

トランジスタ技術

SPECIAL

特集 パソコン周辺機器インターフェース詳解

セントロニクス/RS-232C/GPIB/SCSIを理解するために

No.9



エレクトロニクスの基礎と実用技術を濃縮したフィールド・ワーク・マガジン

トランジスタ技術 SPECIAL

季刊●B5判●定価：①～③③定価1,570円 ③④～④⑤定価1,631円 ④⑥～④⑨定価1,723円 ⑤⑩～⑤⑦定価1,835円 ⑤⑧以降定価1,840円

①個別半導体素子 活用法のすべて 基礎からマスタするダイオード、トランジスタ、FETの実用回路技術	②③回路デザインのためのPLD最新活用法 PLDのプログラミング法からPALライタの製作まで	④②高速デジタル回路の測定とトラブル解析 ハイスピード・デジタル信号を高周波と捉える
③PC9801と拡張インターフェースのすべて 16ビット・パソコンを使いこなすためのハード&ソフト	④④Cによる組み込み機器用プログラミング 16ビットCPUによるメカトロニクス入門 <在庫僅少>	④③Cによるマイコン制御プログラミング 86系ベリフェラルを中心とした
④C-MOS標準ロジックIC活用マニュアル 実験で作る4000B/4500B/74HCファミリ	⑤⑦ハードディスクとSCSI活用技術のすべて 本格活用のためのハード&ソフトのすべてを詳解	④④フィルタの設計と使い方 アナログ回路のキーポイントを探る
⑤画像処理回路技術のすべて カメラとビデオ回路、パソコンと隔合させる	⑤⑧最新・電源回路設計技術のすべて 3端子レギュレータから共振型スイッチング電源まで	④⑤PC98シリーズのハードとソフト 386&486マシンを使いこなす!
⑥Z80ソフト&ハードのすべて 基礎からマクロ命令を使いこなすまでのノウハウを集大成	⑤⑨マイコン独習Z80完全マニュアル 手作りの原点から実用ソフトの作成まで	④⑥アナログ機能ICとその使い方 民生用AV機器からマルチメディア分野で活躍する
⑧データ通信技術のすべて シリアル・インターフェースの基礎からモデムの設計法まで	⑥⑩ニュー・メディア時代のデータ通信技術 赤外線、無線通信技術からLAN、光ファイバを用いた高速通信技術まで	④⑦高周波システム&回路設計 通信新時代の回路技術とシステム設計
⑨パソコン周辺機器インターフェース詳解 セントロニクス/RS-232C/GPIB/SCSIを理解するために	⑥①基礎からのビデオ信号処理技術 複合映像信号の理解からハイビジョン信号の伝え方まで	④⑧作れば解るCPU ロジックICで実現するZ80とキャスル・マシン
⑩IBM PC&80286のすべて 世界の標準パソコンとマルチタスクの基礎を理解する	⑥②実用電子回路設計マニュアル アナログ回路の設計例を中心に実用回路を詳述	④⑨徹底解説 Z80マイコンのすべて Z80CPUの概要から周辺LSIの活用法、ICEのデバッグまで
⑪フロッピー・ディスク・インターフェースのすべて 需要の急増するFDDシステムの基礎から応用まで	⑥③オプト・デバイス応用回路の設計・製作 光素子を使いこなすための製作ドキュメント	⑤⑩フレッシュアーズのための電子工学講座 電磁気学の基礎から電子回路の設計、製作までをやさしく解説
⑬シミュレータによる電子回路理論入門 コンピュータを使ったアナログ回路設計の手法を理解するために	⑥④つくるICエレクトロニクス 機能ICを使って実用機器を作ろう <在庫僅少>	⑤①データ通信技術基礎講座 RS232Cの徹底理解からローカル通信の実用技術まで
⑭技術者のためのCプログラミング入門 MS-C、Quick C、Turbo Cによるソフトウェア設計のすべて	⑥⑤C言語による回路シミュレータの製作 Quick Cでのプログラミングとフィルタ回路の解析	⑤②ビデオ信号処理の徹底研究 映像信号の基礎から高画質化のためのデジタル信号処理の方法まで
⑮アナログ回路技術の基礎と応用 計測回路技術のグレードアップをめざして <在庫僅少>	⑥⑥基礎からの電子回路設計ノート トランジスタ回路の設計からビデオ画像の編集まで	⑤③パソコンによる計測・制御入門 研究室や実験室に必要なデータ収集のノウハウを基礎から解説
⑯A-D/D-A変換回路技術のすべて アナログとデジタルを結ぶ最新回路設計ノウハウ	⑥⑦実用電子回路設計マニュアルⅡ 豊富な回路設計例から最適設計を学ぼう	⑤④実践パワー・エレクトロニクス入門 パワーMOS FETとIGBTの使い方をやさしく解説
⑰OPアンプによる回路設計入門 アナログ回路の誤動作とトラブルの原因を解く	⑥⑧Z80システム設計完全マニュアル 周辺I/Oボードの設計とマイコン・システムの開発	⑤⑤作ってわかる電子回路製作入門 やさしい電子工作からパソコンを使ったシステム開発まで
⑲PC9801計測インターフェースのすべて オリジナル拡張ボードでパソコンを実践活用しよう	⑥⑨A-Dコンバータの選び方・使い方のすべて アナログ信号をデジタル処理するための基礎技術	⑤⑥電子回路シミュレータ活用マニュアル アナログ回路解析だけでなくデジタル回路解析も追加された
⑳アナログ回路シミュレータ活用術 ゲーム感覚の回路設計を体験しよう	④⑩電子回路部品の活用ノウハウ 機器の性能と信頼性を支える受動部品の使い方	⑤⑦最新・スイッチング電源技術のすべて 効率とノイズを重点的に解説したソフト・スイッチングの指南書
㉒デジタル回路ノイズ対策技術のすべて TTL/CMOS/ECLの活用法と誤動作/トラブルへの処方	④①実験で学ぶOPアンプのすべて 汎用OPアンプから高性能OPアンプまで	⑤⑧基本・C-MOS標準ロジックIC活用マスタ 低電圧動作とドライブ能力の向上をはかった

CQ出版社 170 東京都豊島区巣鴨1-14-2 販売部 ☎03-5395-2141 振替 00100-7-10665

定価は1997年4月1日現在の消費税5%を含んだ表示です

トランジスタ技術 SPECIAL

No.9

CONTENTS

FEATURES

セントロニクス / RS-232C / GPIB / SCSIを理解するために
パソコン周辺機器インターフェース詳解

セントロニクス・パラレル

- §1-1 セントロニクス・インターフェースの基礎 ● 里 和政……………2
- §1-2 パラレル・インターフェース用LSIの使い方
● 里 和政/神崎康宏/斉藤健司…………8
- §1-3 セントロニクス↔RS-232C変換器の製作 ● 土屋 哲/矢吹真人…………22
- §1-4 簡易型プリンタ・バッファの製作 ● 斉藤洋司/生沼守英…………28

RS-232Cシリアル

- §2-1 RS-232Cインターフェースの基礎 ● 里 和政……………37
- §2-2 シリアル・インターフェース用LSIの使い方
● 里 和政/相良富美/斉藤健司…………43
- §2-3 PC9801によるBSC伝送の実際 ● 沖野 新……………56
- §2-4 RS-232Cチェッカの製作 ● 鶴野和孝……………67

GPIB

- §3-1 GPIBインターフェースの基礎 ● 里 和政……………74
- §3-2 GPIBコントロールLSIの使い方 ● 里 和政/松井雅行/竹尾佳己…………80
- §3-3 GPIBラインを光ファイバで延長する ● 鶴野和孝……………95
- §3-4 HP9816-PC9801間をGPIBを用いてファイル転送する
● 亘 慎一…………105

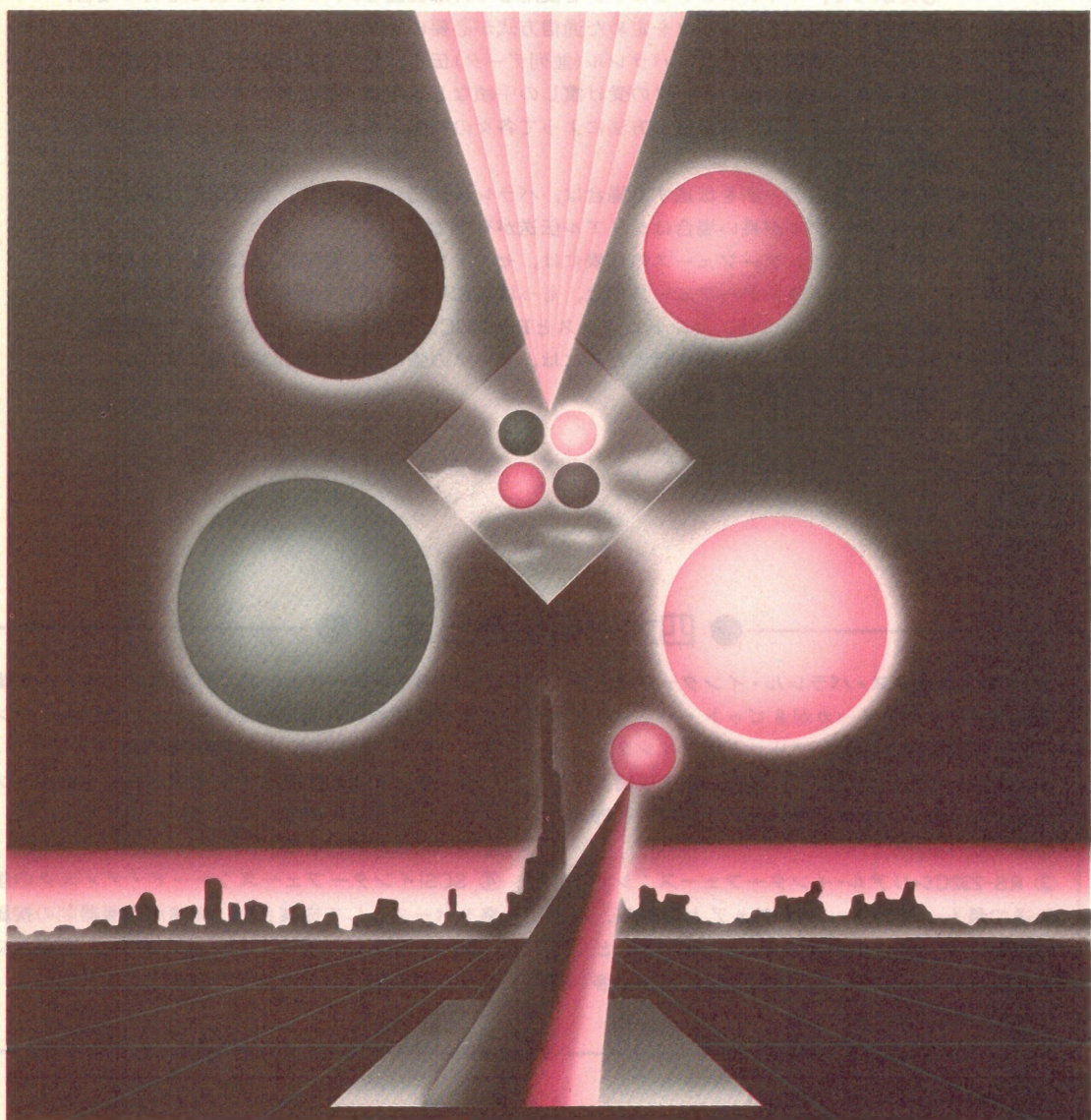
SCSI

- §4-1 SCSIインターフェースの基礎 ● 里 和政……………118
- §4-2 SCSIコントロールLSIの使い方 ● 里 和政/清水哲夫…………124
- §4-3 PC9801用SCSIアダプタの製作 ● 里 和政……………136
- §4-4 NCR53C80評価用ボードの製作 ● 清水哲夫……………148

FEATURES

現在、パソコン/マイコンの周辺機器とのインターフェースとして、セントロニクスや RS-232C、GPIBが多く使われ、将来は高速データ転送が可能な SCSI も利用されそうです。本誌では、それぞれの規格と使われる IC の説明を行ったのち、実際の応用例を示し理解を深めます。

セントロニクス/RS-232C/GPIB/SCSIを理解するために パソコン周辺機器インターフェース詳解



● はじめに

私たちが、他人と会話する場合、お互いの話を確認しながら会話を進めていきます。もしも、同時に話した場合、お互いになにを言っているのかわからず、会話にならなくなります。このようなことが、コンピュータと外部装置の間でも起こります。

コンピュータと外部装置では、次のような会話になります。

コンピュータ：「データを送ります。いいですか。」

外部装置：「データを送ってください。」

コンピュータ：「データを送っています。」

外部装置：「データを受け取りました。」

となります。

この一連の手順を、ハンドシェイクと呼びます。

したがって、マイクロコンピュータを使用して外部装置とデータの受け渡しを行う場合、互換性のあるハンドシェイクを定めた通信方式が必要になります。

また、その通信方式には、パラレル(並列データ)伝送、シリアル(直列データ)伝送の2種類が考えられますが、データの受け渡しの手順なども考慮する必要があります。

ハンドシェイクは、各伝送の方法によって各々の特徴があるため、もっとも適切な方法を選択します。

高速で大量のデータを伝送する場合は、パラレル伝送が有効です。伝送速度は遅くなりますが、ケーブルが長い場合はシリアル伝送が有効です。

パラレル・インターフェースの代表には、セントロニクス、GBIP、SCSIなどがあります。シリアル・インターフェースでは、もっぱらRS-232Cがよく使用されています。

最近のパソコンには、セントロニクスとRS-232Cが標準装備になっています。実際、これらによってデータ伝送を行うためには、データの受け渡しの規則が必要です。

以下それらの説明をします。

● 四大規格のあらまし ●

① セントロニクス・パラレル・インターフェース：米国のプリンタ・メーカーが8ビット・パラレル・データに制御信号を加え、プリンタ用のインターフェース規格としたのが始まり。現在、標準規格と呼ばれるものではなく、各社は準拠という独自の社内規格を作っている。

② RS-232Cシリアル・インターフェース：シリアル伝送で最も一般的なハードウェア上の規格。現在EIA-232-Dという名前になって、RS-232Cとは若干の変更がなされている。本誌では、利用の多い従来からのRS-232Cを用いる。

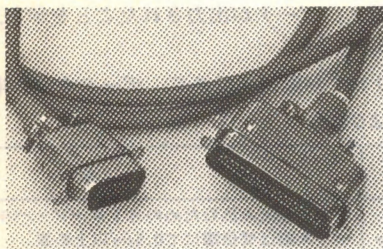
③ GPIBインターフェース：計測用のシステムを組むときに使われる双方向8ビット・パラレル・インターフェース。IEEE-488が標準規格となっており、ハードウェア上の接続は十分考慮されているが、ソフトウェア面で利用者により違いが出てくるので、互換性には注意が必要。

④ SCSIインターフェース：ハード・ディスクの規格をベースとして提案され、高速周辺機器間の接続に使われる新しい規格。ANSIのX3T9.2委員会により、より細かいコマンドの制定もなされている。

§ 1-1

セントロニクス・インターフェース の基礎

里 和政



セントロニクス・インターフェースは、米国セントロニクス社が自社のプリンタ用に開発したコンピュータからプリンタにデータを送るための規格で、安価でかつ高速のデータを送ることができます。現在のプリンタは、ほとんどこのセントロニクスが標準となっています。

このインターフェースの信号線は、すべてTTLレベルで入出力されています。基本的な制御は、3本で行われ、低速または高速にかかわらずどのようなプリンタでも、CPUと同期を取ることができます。

● セントロニクス・コネクタのピン配置

図1に、セントロニクス・コネクタのピン配置を示します。基本的な端子は、1ピンから11ピンの信号線です。これらの線は、すべてツイスト・ペア(19ピンから29ピンのグラウンド・ピンとのより線)になっています。

ほかの端子については、各社で適当な信号線を定義しています。

例として、スター精密、エプソン、日本電気のPC-PR系の各社のプリンタについての各ピンを表1、表2、表3に示します。

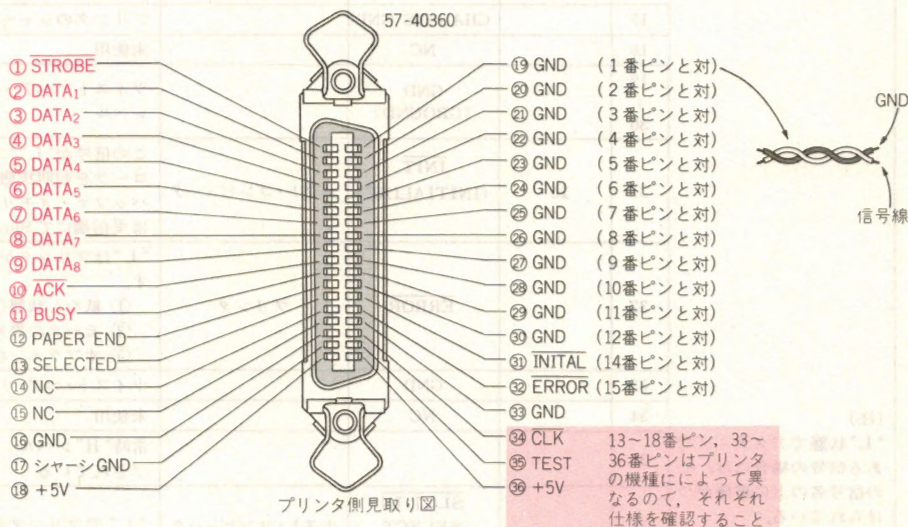
セントロニクスのタイミング

セントロニクスのデータ伝送は、DATA STROBE、ACKNOWLEDGE(ACK)、BUSYの3本の制御線と

〈表1〉⁽¹⁾ スター精密社AR2400のコネクタ・ピンの説明

ピン番号	信号名	入出力区分	ペア・リターン・ピン
1	STROBE	入力	19
2	DATA ₁	入力	20
3	DATA ₂		21
9	DATA ₈		27
10	ACK	出力	28
11	BUSY	出力	29
12	PAPER END	出力	30
13	SELECTED	出力	
16	GND		
17	CHASSIS GND		
18	+5V	出力	
31	INPUT PRIME	入力	
32	ERROR	出力	
33	EXT GND		

〈図1〉
セントロニクス・
コネクタのピン配
置



DATA線によって行います。まず最初に、制御方法について説明しますが、その前に各制御線の意味について、まとめておきます。

► DATA STROBE

DATA線に、データが出力されたことを示す。

► BUSY

現在プリンタが動作中であり、データを受け取れ

〈表2〉⁽²⁾ エプソン社VP130Kのピン配置および説明

	ピン 番号	リターン側 ピン番号	信 号 名	発 信 先	機 能
パラレル・インターフェース信号の場合には、ツイスト・ペア線を使用しリターン側をロジカル・グラウンドに接続する。	1	19	$\overline{\text{STROBE}}$	ホスト・コンピュータ	データを読み込むためのストロブ・パルス。パルス幅は受信端にて0.5 μ s以上必要。定常状態では“H”であり“L”となった後にデータを読み込む
	2	20	DATA ₁	ホスト・コンピュータ	各信号はパラレル・データの1ビット目から8ビット目までの情報を表す。“H”はデータが1であり“L”はデータが0であることを示す
	3	21	DATA ₂		
	4	22	DATA ₃		
	5	23	DATA ₄		
	6	24	DATA ₅		
	7	25	DATA ₆		
	8	26	DATA ₇		
	9	27	DATA ₈		
	10	28	$\overline{\text{ACKNLG}}$ (ACKNOWLEDGE)	プリンタ	“L”はプリンタが次のデータを受け付ける用意ができていることを示す。パルス幅は約11 μ s
	11	29	BUSY	プリンタ	“H”はプリンタがデータを受け取れないことを示す。逆に“L”はプリンタがデータを受け取れることを示す。この信号が“H”になるのは次の場合、 ① データ・エントリ中 ② 印字中およびヘッド・キャリア動作中の一部の時間 ③ 紙送り中の一部の時間 ④ エラー状態 ⑤ オフライン状態
	12	30	PE (PAPER END)	プリンタ	“H”はプリンタに用紙がないことを示す
	13		SLCT (SELECT)	プリンタ	常時“H”レベル。3.3k Ω で+5Vにプルアップされている
	14		$\overline{\text{AUTO FEEDXT}}$	ホスト・コンピュータ	この信号が“L”になると、プリンタは印刷終了後、自動的に1行の改行を行う
	15		NC		未使用
	16		GND		ロジック供給電源の0Vレベル
	17		CHASSIS GND		プリンタのシャーシのGNDレベル
	18		NC		未使用
	19 ↓ 30		GND (GROUND)		ツイスト・ペア・リターン用信号グラウンド・レベル
	31	16	$\overline{\text{INIT}}$ (INITIALIZE)	ホスト・コンピュータ	この信号が“L”になると、プリンタ・コントローラを初期状態にリセットし、プリント・バッファ・メモリがクリアされる。パルス幅は受信端にて50 μ s以上必要
	32		$\overline{\text{ERROR}}$	プリンタ	“L”はプリンタがエラー状態にあることを示す。 ① 紙なし状態(ESC8で解除される) ② モータの異常動作 ③ オフライン状態
	33		GND		ツイスト・ペア・リターン用グラウンド
	34		NC		未使用
	35				常時“H”レベル。3.3k Ω で+5Vにプルアップされている
	36		SLCT IN (SELECT INPUT)	ホスト・コンピュータ	“L”でプリンタを選択する

(注)

“L”状態でアクティブである信号の場合には、その信号名の上に横棒がつけられている。

ないことを示す。

(または、データ・バッファがフルである)

▶ACK

データの読み取りが正常に終了した。

セントロニクス社のタイミングも各社によって差があります。セントロニクス・インターフェースのタイミングを図2、図3、図4に示します。

制御の基本的なタイミングは、図5のようになります。タイムチャートを見るとわかるように、BUSY信

号と同じタイミングでACK信号もインアクティブとなるために、3線ハンドシェイクでなくても、2線ハンドシェイクで動作できます。

2線ハンドシェイクは、DATA STROBEとACKまたはBUSYのどちらか一方を使用します。一般的には、DATA STROBEとBUSYがよく使用されています。

最近のパソコンでは、2線ハンドシェイクを採用しているものが多くあります。参考例に、PC9801のセ

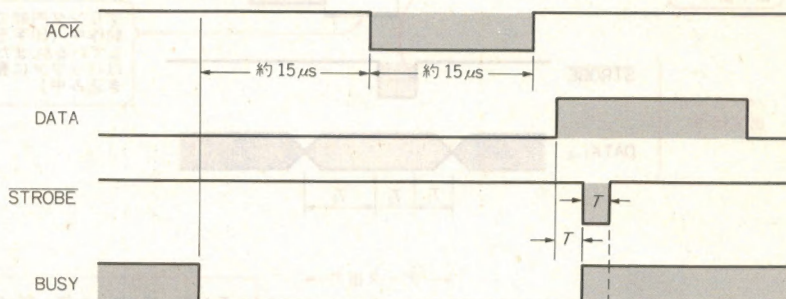
〈表3〉⁽³⁾

日本電気製PC-PR201H2
のピン配置

ピン番号	信号名	入出力区分	ペア・リターンピン	備考
1	DATA・STB	入力	19	
2	DATA ₁	入力	20	
3	DATA ₂		21	
9	DATA ₈		27	
10	ACK	出力	28	
11	BUSY	出力	29	
12	PE	出力	30	Paper End (紙切れ)
13	SELECT	出力		プリンタのセレクト(オンライン)
16	SG			Signal Ground
17	FG			Frame Ground
18	+5V	出力		
31	INPUT PRIME	入力		初期化(プリンタを初期化する)
32	FAULT	入力		エラー
33	SG			グラウンド

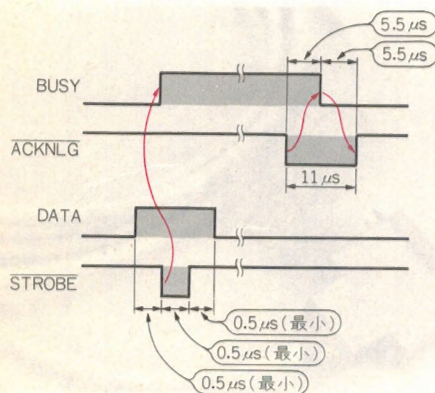
〈図2〉⁽¹⁾

スター精密製AR2400
のタイムチャート

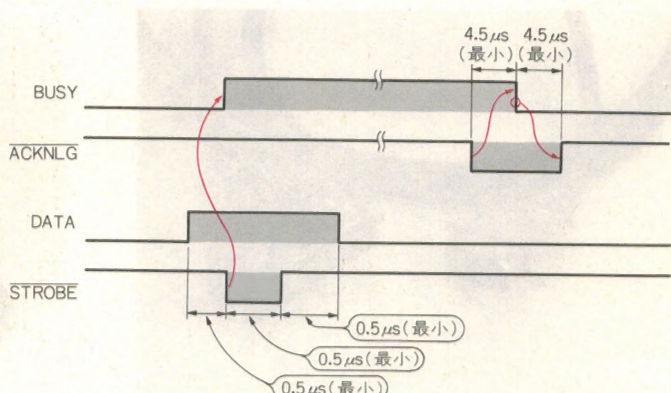


T: 0.5μs 以上

〈図3〉⁽²⁾ エプソン製プリンタのタイムチャート



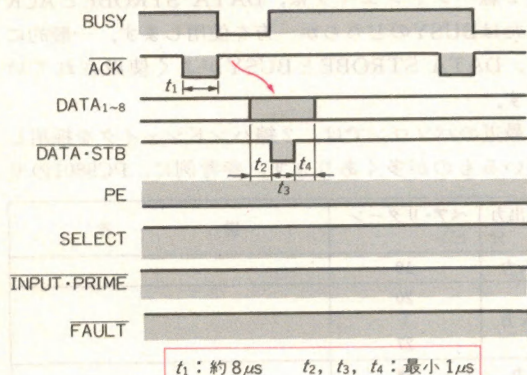
(a) VP130K



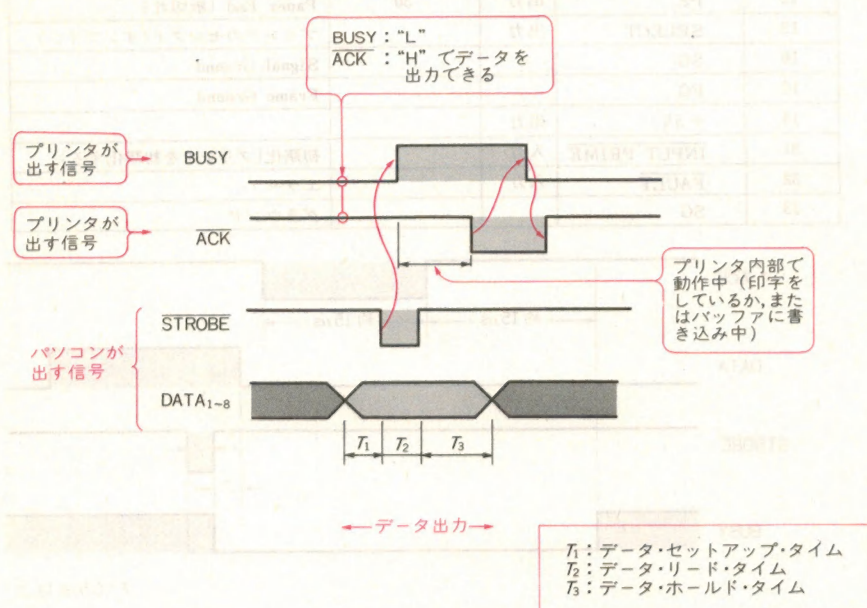
(b) IP130K

セントロニクス・コネクタの配置を図6に示します。この場合、ACK線には何も接続しません。

〈図4〉⁽³⁾ 日本電気製PC-PR201H2/460LP/601のタイムチャート

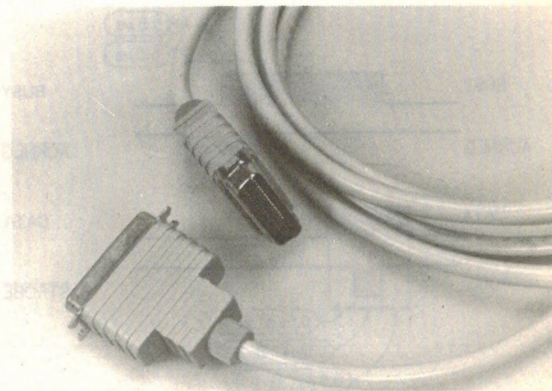
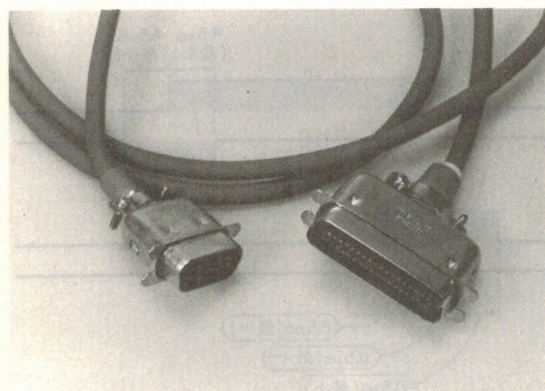
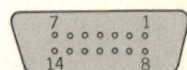


〈図5〉 セントロニクスの基本タイムチャート



〈図6〉 PC9801のセントロニクス用プリンタ・インターフェースのピン配置

ピン番号	信号名
1	PSTB
2	PDB ₀
3	PDB ₁
4	PDB ₂
5	PDB ₃
6	PDB ₄
7	PDB ₅
8	PDB ₆
9	PDB ₇
10	NC
11	BUSY
12	NC
13	NC
14	GND



〈写真1〉 プリンタ・コネクタの外観

用語解説

● ツイスト・ペア

プリンタのケーブルのように複数の信号線がある場合、お互いの線同士が干渉合っってノイズが乗らないように、信号線ごとにグラウンド線をねじり合わせたものです。

● プリンタ・データ・バッファ

プリンタの印字速度は、CPUからのプリント・データの転送速度よりもかなり遅いため、CPUがプリンタを待つことになります。CPUはその間ほかの仕事ができないため、プリンタ自身に数10Kバイトのメモリをもたせて、プリント・データを先に取り込むようにしたものです。このメモリをプリンタ・データ・バッファと呼びます。

● タイミング

装置などを制御する場合に、各制御信号を有効にする時間間隔のことです。

● インアクティブ

制御線が“L”か“H”かを示すときに用いるが、

通常アクティブを真(有効)としてインアクティブを偽(無効)とします。制御線が負論理の場合は、アクティブを“L”、インアクティブを“H”とします。正論理の場合は、その逆になります。

● 8080系

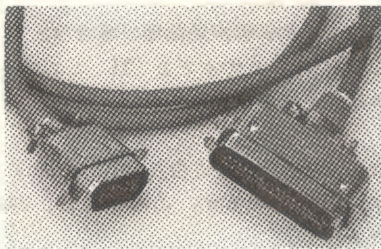
インテル社の8080 CPU用に合うバス・タイミングで作られたLSIをいいます。この中には、Z80、8086なども含まれます。

● I/Oウェイト

CPUが、周辺LSIに対して入出力を行う場合に、LSIのI/Oタイミング(アクセス・スピード)が遅いために、アクセスが終わるまでCPUが待つことをいいます。

● モード

周辺LSIには、汎用性をもたせるため多くの機能があり、プログラムで指定することによって選択することができます。この機能のことをモードと呼んでいます。



§ 1-2

パラレル・インターフェース用 LSIの使い方

里 和政/神崎康宏/斉藤健司

8255A (PPI)

8255Aは、8080系のCPUでよく利用される、プログラマブル・パラレルI/Oコントローラであり、8/16ビットCPUの区別なしに使用されています。32ビットCPUでも使用されるでしょう。

このLSIは、プログラムによってコントロールすることができます。

● 8255Aの内部構成

8255Aの内部ブロック図を図1に、図2に端子配置図を示します。I/Oポートは、A、B、Cと三つあります。A、B、Cと三つに分けると同時に、AとCの半分、BとCの半分という、二つのグループに分けて考えることもできます。各モードに応じて、A、Bはそれぞれ独立して機能します。Cは、モード0以外は4ビットず

つA、Bのそれぞれのグループとともに、コントロール信号としての機能を果たします⁽¹⁾。

● モードの使い方

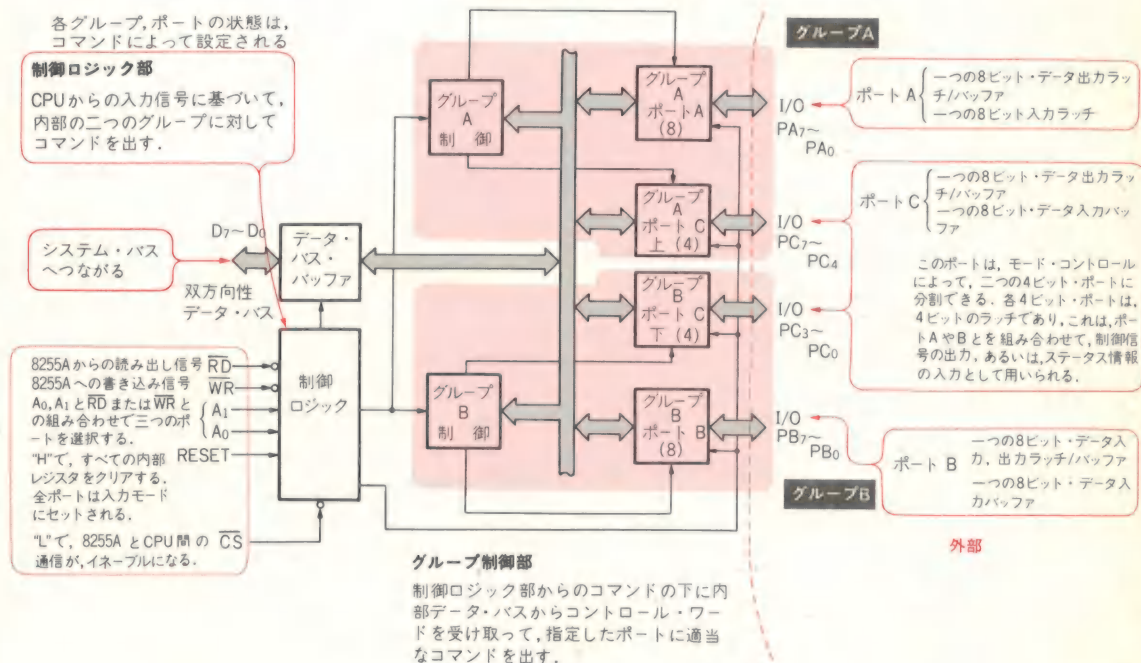
この素子の最もシンプルな使用法は、**モード0のたんなる入出力のみを行うポートとして利用する場合**です(図3、図4)。

この場合、入力動作ではその入力時にA、B、Cの各ポートに加わっているデータがCPUに読み込まれます。出力動作では、各ポートへ出力されたデータは、出力動作後も同じデータが出力され続けます。これは、各ポートへの書き出し時に出力データが保持されるためです⁽²⁾。

● 8255Aのタイミング上の問題点

本来このLSIは、8ビット用に設計されているために、16/32ビットなどの高速のCPUに対しては、アクセス・タイムなどで問題が発生することもあります(高

〈図1〉⁽¹⁾ 8255Aのブロック図



速用のLSIもある)。

そのため、CPUがI/Oをアクセス時に、I/Oウェイトを入れる必要があります。また、リセット直後、I/O素子にコマンド・ライト後、内部動作を行うために、若干のソフトによるウェイトが必要です。

これらの点については、マニュアルをよく読むことです。実際使ってから、わかることもあります。

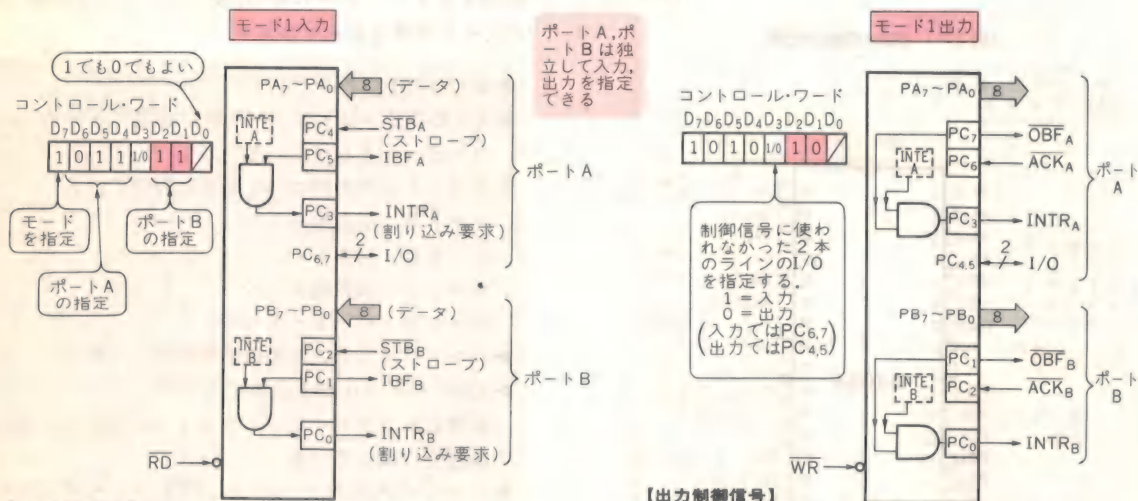
■ セントロニクスへの応用

この8255Aを使用して、セントロニクス・インターフェースを考えてみます。ハンドシェイクは、2線式とします。

● モードの設定手順

ここでの8255Aのモードは、モード0を使用します。

〈図3〉⁽⁷⁾ 8255Aモード1の説明



【入力制御信号】

STB (Strobe Input):

このピンへの“L”入力データが入力ラッチへ入る。

IBF (Input Buffer Full FF):

STBに対する応答信号。データが入力ラッチに入ったことを“H”で示す。STBが“L”になることでセットされ、RD入力の立ち上がりでリセットされる。

INTR (Interrupt Request):

CPUに対する割り込み要求信号として用いるもので、“H”になる。STB=1, IBF=1, INTE=1のときセットされ、RDの立ち上がりでリセットされる。

内部割り込みマスク・フリップフロップの制御は、

▶ INTE_A—ビット・セット/リセットによるPC₄の制御

▶ INTE_B—ビット・セット/リセットによるPC₂の制御

【出力制御信号】

OBF (Output Buffer Full FF):

CPUが指定したポートにデータを書き込んだとき、このピンが“L”になる。このフリップフロップ(FF)は、WR入力の立ち上がりでセットされ、ACKが“L”になることでリセットされる。

ACK (Acknowledge Input):

CPUが出力したポートAまたはポートBのデータをI/O側で受け取ったとき、8255Aに対してその応答信号である“L”を出力する。

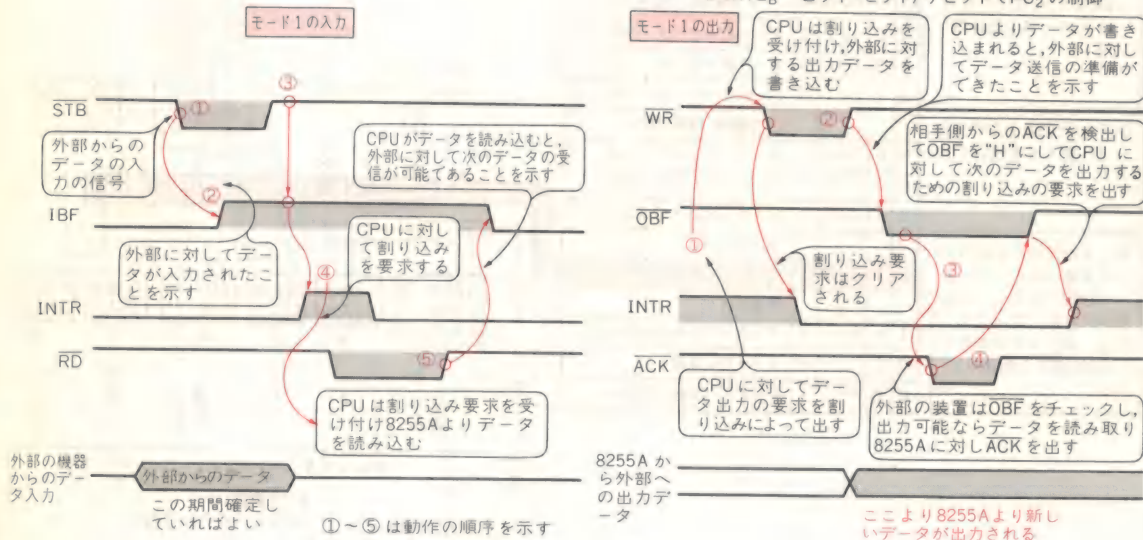
INTR (Interrupt Request):

CPUに対する、I/Oからのデータ転送終了割り込み信号で“H”が出る。この信号は、OBF=1, INTE=1のときに、ACKによってセットされ、WRの立ち上がりでリセットされる。

内部割り込みマスク・フリップフロップの制御は、

▶ INTE_A—ビット・セット/リセットでPC₆の制御

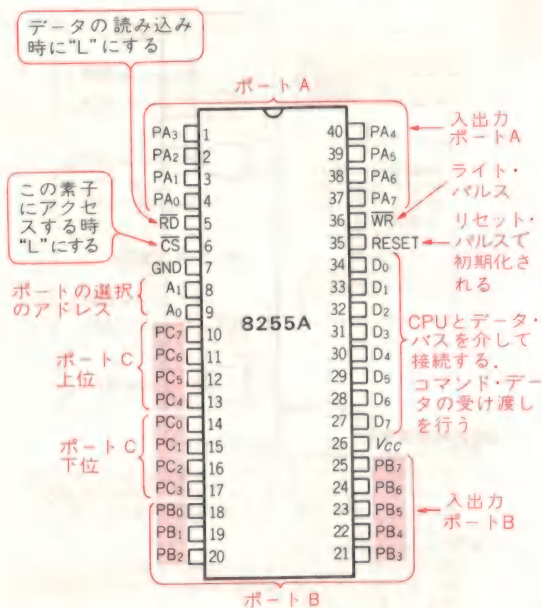
▶ INTE_B—ビット・セット/リセットでPC₂の制御



データの出力用にはAポート、BUSY信号の入力用にCポートの下位4ビットのうちの1ビットを割り当てます。STROBE信号の出力用にCポートの上位4ビットのうちの1ビットを割り当てます。図5に回路図、図6にCポートの割り当てを示します。

8255Aには、Cポートに対してのみ、特定のビットを単独でON/OFFできる機能があるため、容易にSTROBE信号を出力することができます。

〈図2〉⁽⁷⁾ 8255Aの端子配置



BUSY信号の状態は、Cポートを読み出すことでわかります。

図7にフローチャートを示します。参考までに8086でのプログラムをリスト1に示します。

Z80 PIO

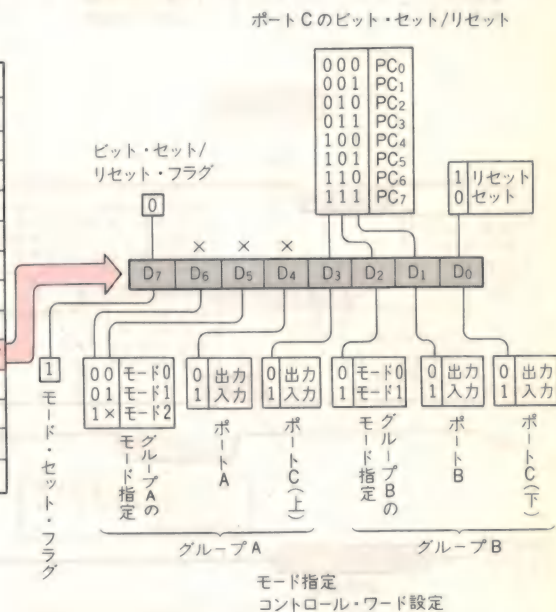
Z80 PIOは、Z80ファミリの汎用8ビットの並列入出力インターフェース用のLSIです。このLSIは、次のような特徴をもっています。

- ▶ 40ピンDIPである
- ▶ AとBの二つの入出力を任意に設定できる8ビット・ポートをもっている
- ▶ 各ポートは次の四つのモードに設定できる
 - モード0：出力モード
 - モード1：入力モード
 - モード2：双方向モード
 - モード3：ビット・モード
- ▶ ハンドシェイクのための信号線を二つもっている
- ▶ Z80のモード2割り込みのために、ベクトルの生成機能およびデジィ・チェーンの割り込み処理機能を内蔵している
- ▶ すべての入出力ラインは、TTLコンパチブルとなっている。またポートBは、ダーリントン・トランジスタなどの電流容量の大きい負荷を直接ドライブする能力をもっている。

Z80独自の機能をフルに利用しようとするとき、Z80ファミリの周辺インターフェース用のデバイスを

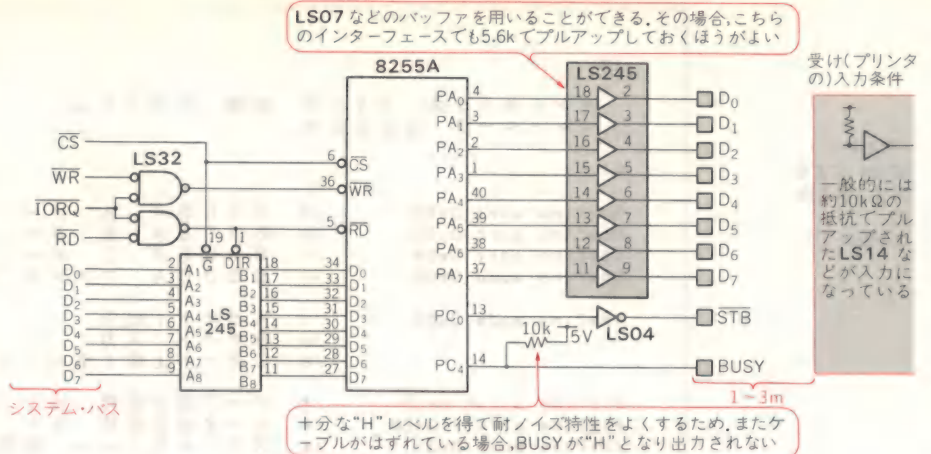
〈図4〉⁽⁷⁾ 8255Aのコントロール・ワードの設定

ポート指定		CS	A ₁	A ₀	WR	RD	入力動作 (READ)
IN命令で、A, B, Cの各ポートのデータを読み取れる		0	0	0	1	0	ポートA→データ・バス
		0	0	1	1	0	ポートB→データ・バス
		0	1	0	1	0	ポートC→データ・バス
OUT命令で、A, B, Cの各ポートへデータを出力できる		0	0	0	0	1	データ・バス→ポートA
		0	0	1	0	1	データ・バス→ポートB
		0	1	0	0	1	データ・バス→ポートC
モード設定のためのコマンドを書き込む。このモード設定で各ポートの状態が決まる。D ₇ =0のときは、ポートCの各ビットのON/OFFの制御がでる							コントロール
		0	1	1	0	1	データ・バス→コントロール・レジスタ
							機能なし
		1	x	x	x	x	データ・バス→3ステート
		0	1	1	1	0	イリガル状態
		0	x	x	1	1	データ・バス→3ステート



〈図5〉(3),(7)

8255Aを使ったセントロニクス・インターフェース回路図



(a) 入力回路

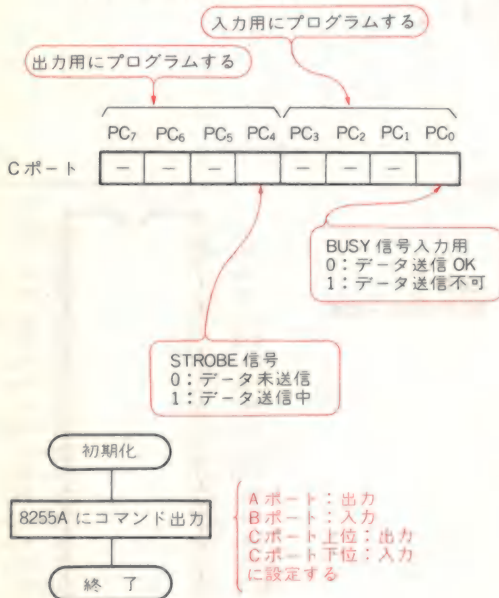
信号名	回路型式
DATA _{1~8}	 7414 相当品
DATA・STB	 7414 相当品
INPUT・PRIME	 7414 相当品

(b) 出力回路

信号名	回路型式
ACK, FAULT BUSY, PE SELECT	 7406
+5V	

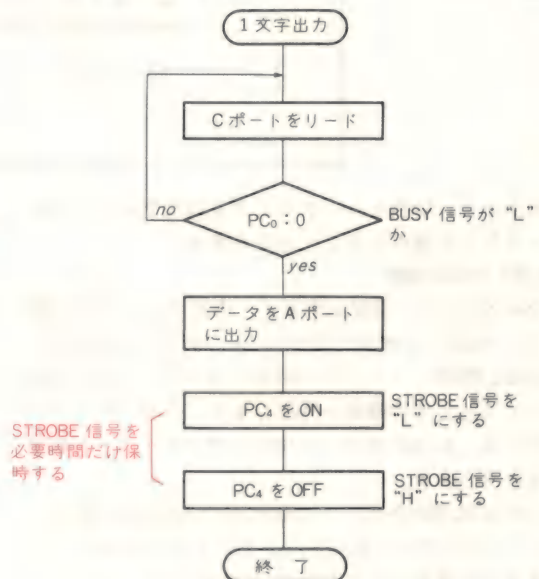
(*PC-PR406LP より)

〈図6〉8255A Cポートの割り当て



(a) 8255A の初期化

〈図7〉8255Aを使ったセントロニクス制御のフローチャート



(b) 1文字出力

〈リスト1〉
8255A による
プリンタ入出力
(Lattice C)

```

/*
    セントロニクス プリンタ 制御 プログラム
    コントローラ 8 2 5 5 A

*/

#define aprt 0x00 /* 8 2 5 5 A A ポート */
#define bprt 0x02 /* 8 2 5 5 A B ポート */
#define cprt 0x04 /* 8 2 5 5 A C ポート */
#define pcmd 0x06 /* 8 2 5 5 A コマンド ポート */

#define mode 0x83 /* A ポート: 出力 */
/* B ポート: 入力 */
/* C ポート上位: 出力 下位: 入力 */

#define stron 0x09 /* データ出力信号 オン */
#define stroff 0x08 /* データ出力信号 オフ */
#define busy 0x01 /* プリンタビジー 信号 */

/*
    8 2 5 5 A 初期化 ルーチン
*/
pinit()
{
    outp(pcmd,mode); /* モード 設定 */
    return(0);
}

/*
    プリンタ 1文字出力 ルーチン
*/
pout(data)
char data;
{
    int i;

    /* プリンタのビジー チェック */
    while(inp(cprt)&busy){ }

    /* 1文字出力 */
    outp(aprt,data);
    /* ストロープ オン */
    outp(pcmd,stron);
    for(i=0;i==2;i++){ }
    /* ストロープ オフ */
    outp(pcmd,stroff);

    return(0);
}

```

使用すると、特別なハードウェアを追加することもなくシステムを構成することができます。

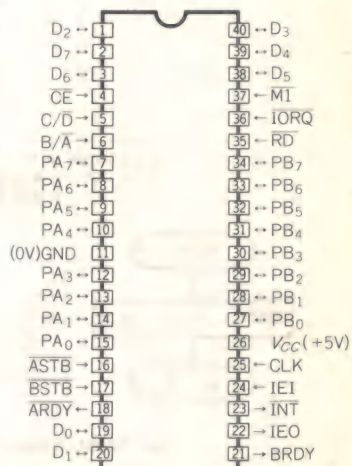
● Z80 PIOの構成

Z80 PIOのピン配置を図8に、内部ブロック図を図9に、各端子の機能を図10に示します。Z80 PIOも8255Aと同様にコマンドの設定によって、初めてI/Oデバイスとしての機能を発揮します。このコマンドの設定には、A,Bの各ポート用のコマンド・ポートが用意されています。

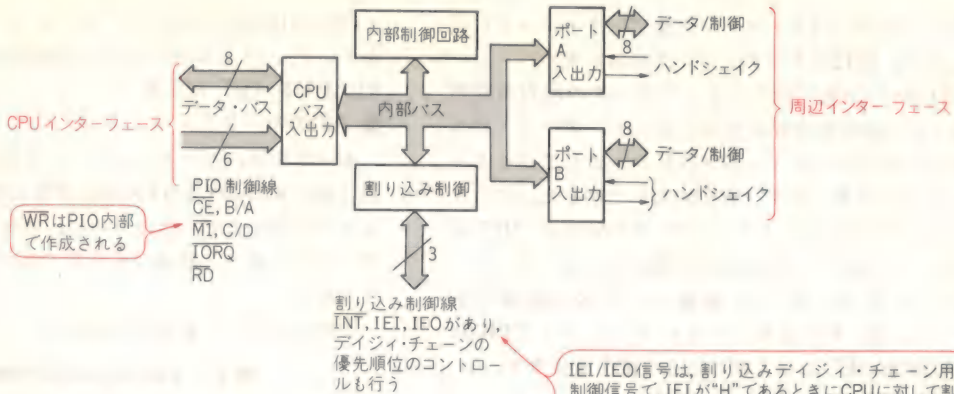
これはA,Bの各ポートの選択用の入力端子(6番ピン)と、それぞれのポートのコマンドであるかデータであるかを選択する入力端子(5番ピン)の二つによって、各ポートの選択が制御されます。

具体的には、これらの入力端子にアドレス・バスの

〈図8〉
Z80 PIOのピン配置



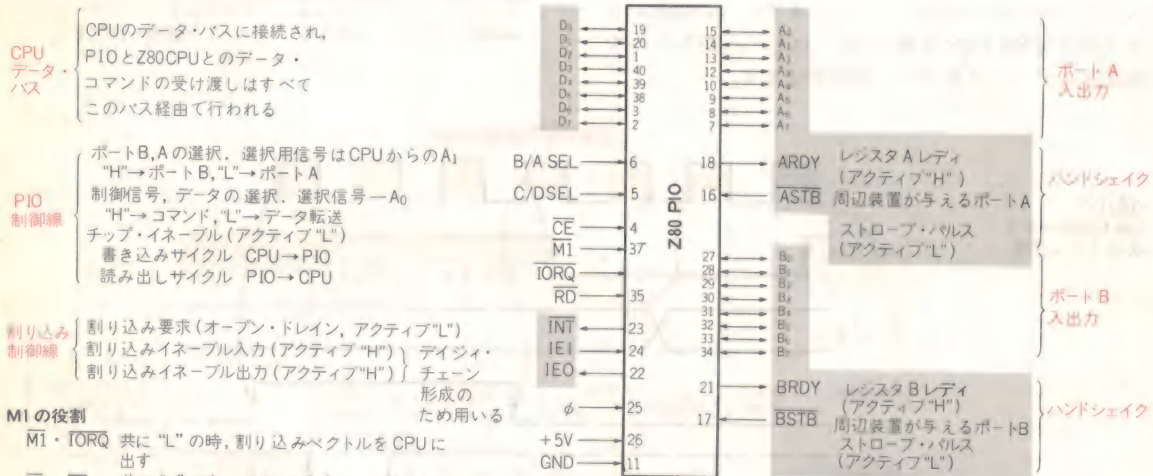
＜図9＞⁽⁷⁾
Z80 PIOの内部
ブロック図



IEI/IEO信号は、割り込みデジジィ・チェーン用の制御信号で、IEIが“H”であるときにCPUに対して割り込みを行える。IEOは、IEIが“L”またはPIOが割り込み待ちのときに“L”になる。それ以外では、“H”となっている。

$\overline{\text{INT}}$ は、CPUの $\overline{\text{INT}}$ に接続して割り込みがあることを知らせる信号。

〈図10〉⁽⁷⁾ Z80 PIOの機能説明



ハンドシェイク信号は、モードによって意味が異なる

ハンドシェイク信号は、モードによって意味が異なる	
RDY (READY) 出力モード: 周辺装置へのデータ転送が準備できたことを示すため、アクティブとなる 入力モード: 入力レジスタが空となり、周辺装置からのデータ受け入れ準備ができた時アクティブとなる 双方向モード: ARDY-ポート A の出力レジスタ内容の出力準備が完了した時アクティブとなる。BRDY-ポート A のデータ受け入れ準備ができた時アクティブとなる ビット制御モード: 強制的に“L”状態になる	STB (STROBE) 出力モード: 周辺装置がPIO からデータを受信したことを通知する信号。立ち上がりが有効 入力モード: 周辺装置から、入力レジスタへデータをロードした時に出力される。アクティブになった時、データがPIO にロードされる 双方向モード: ASTB がアクティブの時、ポート A の出力レジスタからのデータが、ポート A の双方向データ・バス上にのせられる。信号が立ち上がりれば周辺装置がデータを受け取ったとみなされる。BSTBは、周辺装置からポート A の入力レジスタへのデータのストロブ(書き込み)に用いられる ビット制御モード: 無効

8255Aでもハンドシェイクの必要な場合は、A、Bの2ポートしか使用できません。その場合、割り込み処理の機能を内蔵したZ80 PIOのほうが有利になります。

◆ Z80 PIOのモードの説明

● PIOのモード 0

A₀, A₁を接続すると、表1に示すようにそれぞれに対応したポート・アドレスが設定できます。

モード0は、バイト・データをデータ・ポートに出力します。図11にそのタイミングを示します。

CPUからのOUT命令によってデータの出力を開始します。PIOが出力要求を受け取ると、データ・バス上のデータをデータ・レジスタにラッチしてデータ・ポート上に出力します。WRが立ち上がると、クロックの立ち下がりのタイミングで、READYを“H”にします。これでデータの出力準備ができました。

データを受け取った装置は、STROBE信号を“L”にして、その立ち上がりのタイミングでPIOはREADYを“L”にして割り込みを発生させます。

● PIOのモード1

モード1は、バイト・データを装置から入力します。図12にそのタイミングを示します。

PIOにデータを送りたい装置は、STROBE信号を“L”にすることによりPIOを起動させます。STROBEが“L”になるとデータ・ポートのデータを入力データ・レジスタにラッチします。この信号の立ち上がりでREADY信号を“L”にして、割り込みを発生させデータがあることを知らせます。

CPUがIN命令を出すと、データ・レジスタのデータをデータ・バス上に出力して、RD信号の立ち上がりでREADYを“H”にします。

● PIOのモード2

モード2は、モード0とモード1の組み合わせで、各2組のREADYとSTROBE信号を使ってハンドシェイクを行います。そのために、ポートAだけが双方向となり、ポートBは、ビット・モードでしか使用できません。

このときポートBの割り込みは、ハンドシェイクに

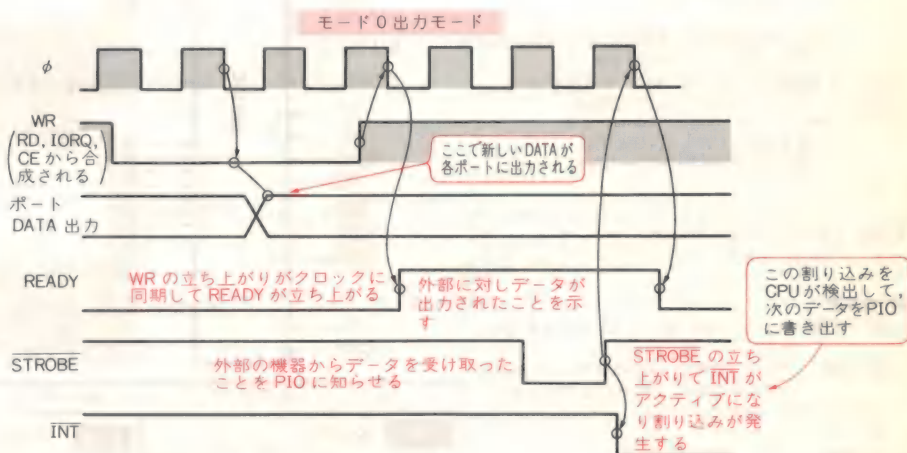
〈表1〉⁽⁷⁾ Z80 PIOとA₀, A₁の接続例

	A ₁	A ₀		A ₁	A ₀
	C/D	A/B		A/B	C/D
Aポート・データ	0	0	Aポート・データ	0	0
Bポート・データ	0	1	Aポート・コマンド	0	1
Aポート・コマンド	1	0	Bポート・データ	1	0
Bポート・コマンド	1	1	Bポート・コマンド	1	1

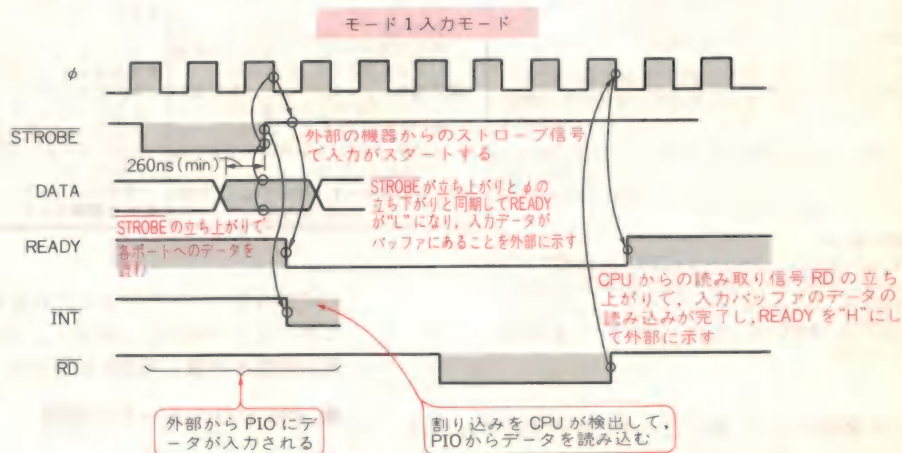
データ、コマンド・ポートが連続してアドレスとなる

それぞれポートのデータ、コマンドが連続したアドレスとなる

〈図11〉⁽⁷⁾
Z80 PIOモード0
のタイミング図



〈図12〉⁽⁷⁾
Z80 PIOのモード1
のタイミング図



使用されるために、割り込みを使わないポーリングで確認する必要があります。

図13にモード2のタイミングを示します。出力のタイミングは、モード0と同様であり、入力タイミングは、モード1と同じです。

● PIOのモード3

モード3はビット・モードと呼ばれ、ハンドシェイクなしに、直接ポート上のデータを入出力します(図14)。

🔍 Z80 PIOのプログラミング

● Z80 PIOの初期設定は数ステップのコマンド設定を必要とする

Z80 PIOの初期設定はA, Bの各ポートに対して、必要に応じて三種類のコマンドを書き込むことで行います。その三種類のコマンドは、次に示すものです(図15)。

(1) 割り込みベクトル

D₀ビットがゼロのコマンドは、割り込みベクトルとみなされます。これはZ80の割り込みを、モード2

で実行する場合に必要になります。

(2) モード設定

Z80 PIOで設定可能な四つのモードを指定するためのコマンドです。D₇, D₆ビットの組み合わせで、図に示すように0から3までのモードが決まります。このコマンドはD₀からD₃の4ビットがともに1となっています(図16)。

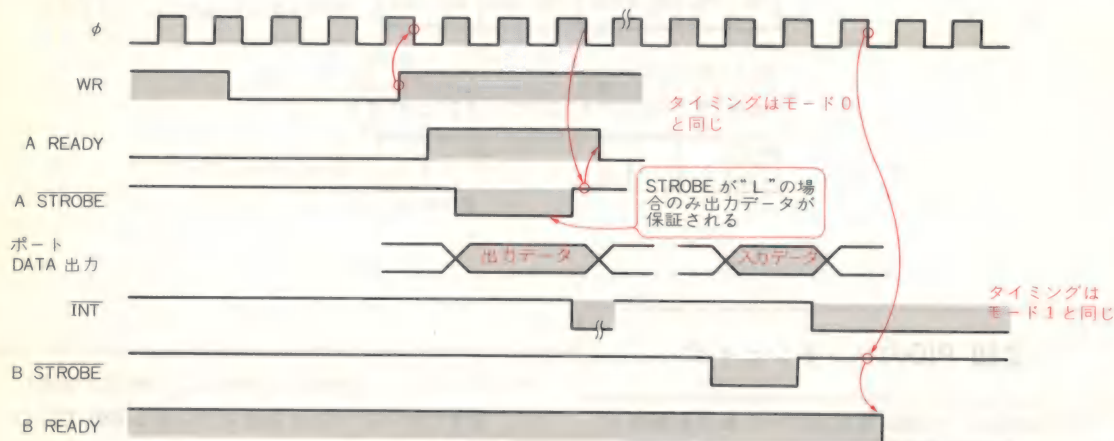
モード3のビット制御モードを指定した場合は、次に各ビットの入出力を決めるためのコマンドを書き込みます。各ビットは1で入力、0で出力になります。

(3) 割り込み制御のコマンド

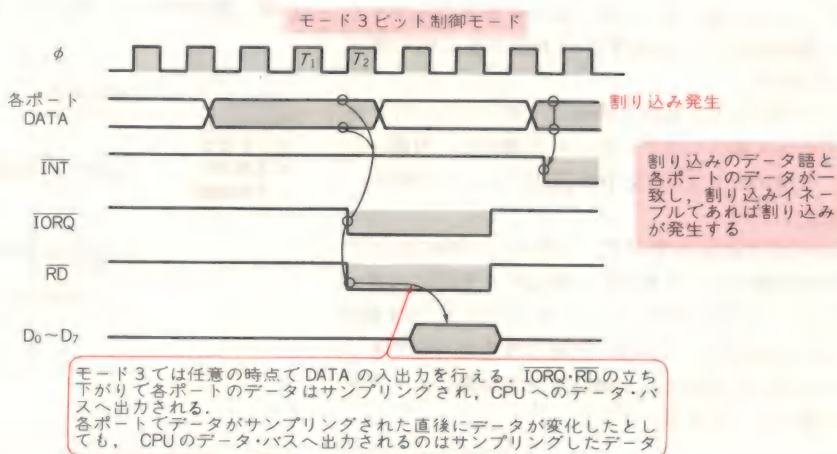
PIOからの割り込み要求の可否を制御するコマンドです。ビット制御モードに対しては、各ビットごとに割り込みの必要の有無を指定することができます。また、割り込みの発生するための条件を、ビット同士のOR、またはANDの関係からも指定することができます。

このコマンドは、図15に示すように下位4ビットが7Hとなっていて、ビット制御モード以外では、割り

〈図13〉 Z80 PIOモード2のタイミング図

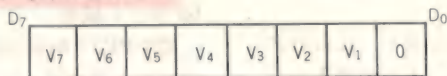


〈図14〉⁽⁷⁾
Z80 PIOのモード3
のタイミング図

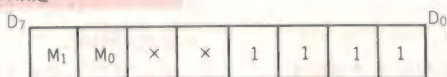


〈図15〉⁽⁷⁾
Z80 PIOの制御語

1. 割り込みベクトルのロード

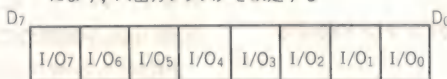


2. モードの設定



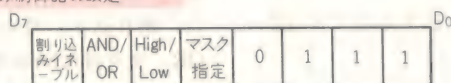
モード3を指定した場合、次に書き込むデータにより、入出力レジスタを設定する

M ₁	M ₀	モード
0	0	モード0 出力
0	1	モード1 入力
1	0	モード2 双方向
1	1	モード3 ビット制御



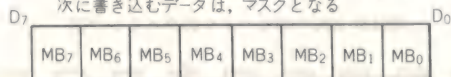
I/O = 1 ならば、入力ビットとなる
I/O = 0 ならば、出力ビットとなる

3. 割り込み制御語の設定



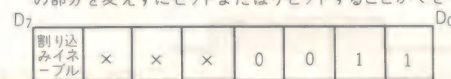
モード3のみで使用

マスク指定 (Mask follows) が“H”レベルならば、次に書き込むデータは、マスクとなる



MB = 0 ならば、モニタ・ビットとなる
MB = 1 ならば、非モニタ・ビットとなる

ポートの割り込みイネーブル・フリップフロップ (IFF) を、次のようなコマンドを使用して、割り込み制御語のほかの部分を変えずにセットまたはリセットすることができる



D₇ = 1 イネーブル
D₇ = 0 ディセーブル

x: どちらでもよい

Z80 PIOのハンドシェイク

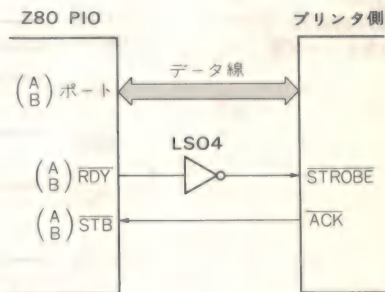
Z80 PIOには、外部装置とのデータ転送が簡単に行えるように、ハンドシェイク機能があります。ハンドシェイクは、ポートAとBのどちらでも使用でき、制御信号としてRDYとSTB線が各2組用意されています。

ハンドシェイクを行う場合は、モード0, 1, 2のいずれかを選択します。モードの選択は、内部レジスタを使用します(本文中の図15「モードの設定」参照)。

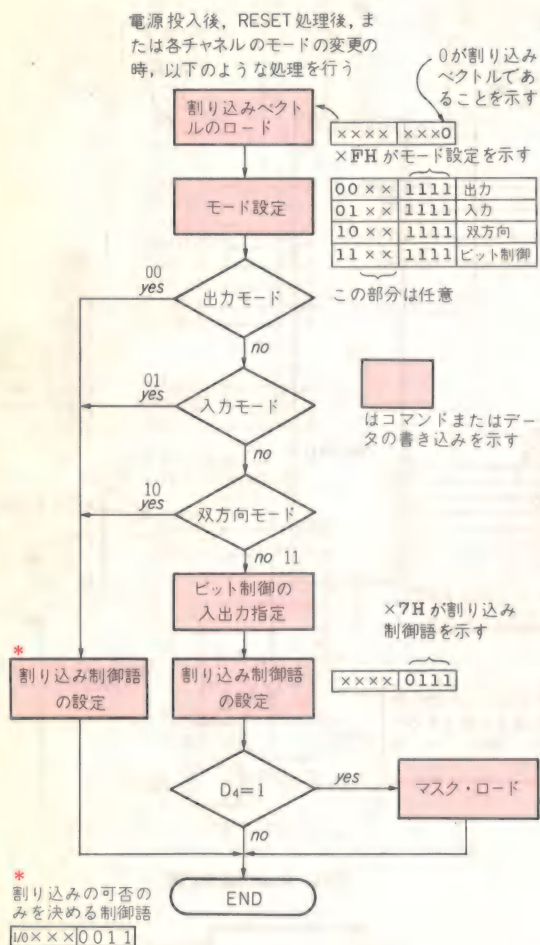
RDY信号は出力信号で、入力モード時にはデータを受け取ったことを示し、出力モード時にはポート上のデータを出力したことを示します。STB信号は入力信号で、入力モード時には入力可能であることを示し、出力モード時には外部装置がデータを受け取ったことを示します。

セントロニクスを制御する場合は、2線式で使用します。図Aにセントロニクス用プリンタとZ80 PIOの接続を、図Bに制御タイミングを示します。この方法が最も簡単ですが、モード2では使用できません。

〈図A〉
Z80 PIO と
セントロニクス用
プリンタの接続



〈図16〉⁽⁷⁾ Z80 PIOのコマンド設定のフローチャート



込み発生の有無の制御のみを行います。

◆ セントロニクスへの応用

Z80 PIOの具体的な使用例として、セントロニクス・インターフェースの回路を考えます。このインターフェースでは、Z80の割り込み機能を利用できるようにしてあります(図17)。

この回路のためのプログラムを、リスト2、リスト3に示します。アセンブラとターボ・パスカルのプログラムを示してあります。

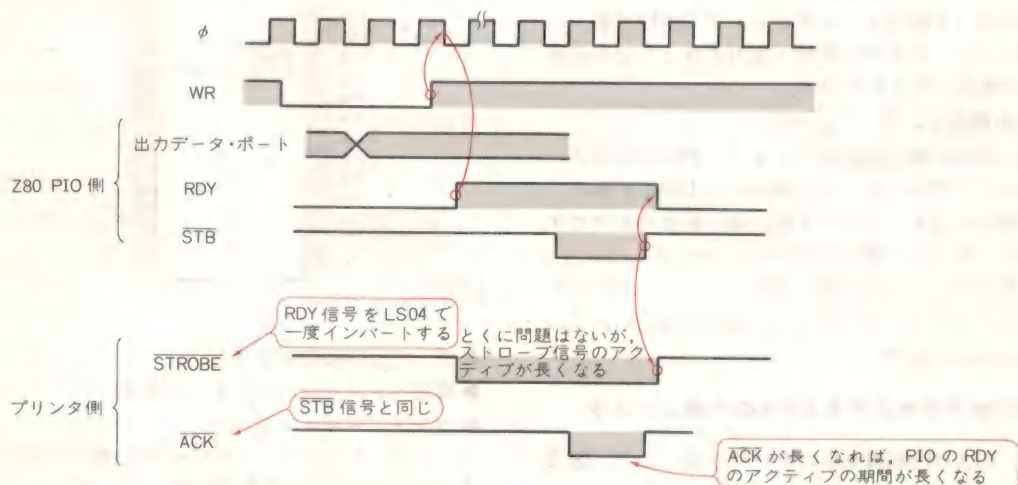
STBはハードウェアで作成してあるので、ソフトウェアの処理は必要ありません。このインターフェースは、STBのパルスを25μsくらいになるようにするため、ハードウェアで作成しました。

しかし最近のプリンタのインターフェースでは、ほとんどがSTBパルスは約1μsの仕様となっています。新しく作るなら図に示すように、B₂を出力端子としてソフトウェアでSTBを作ることでコストが安くなります。

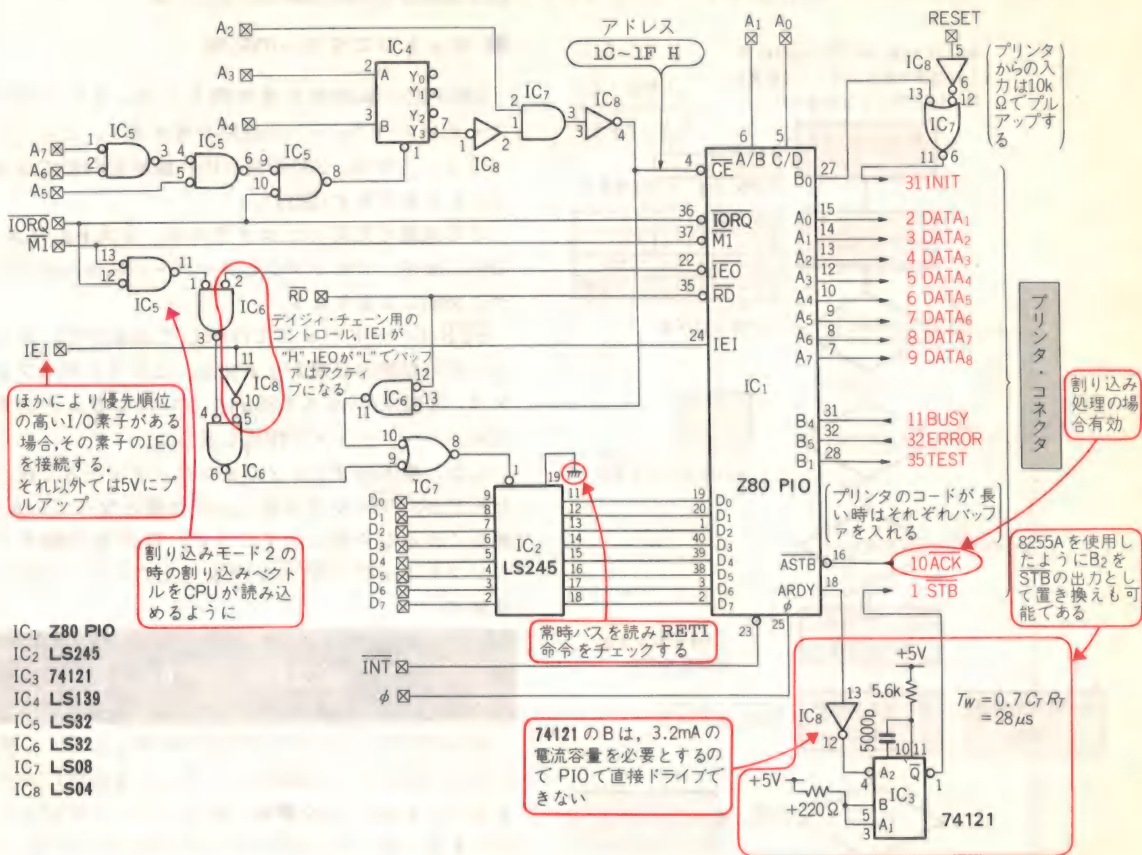
6821/6321 (PIA)

68系の並列入出力用周辺LSIの代表としてよく使われるのがPIAです。低価格でいろいろな応用に適合できるのですが、「少々難解である」という声がよく聞かれます。モトローラ社のオリジナル・データ・シートでは機能と諸特性だけだったのが、日立製のマニュアルではたんなる直訳ではなく、内部の動作原理まで懇切丁寧に独自の書き方で解説、記述してあり、初めて

〈図B〉 図Aの接続によるPIOのタイミング



〈図17〉 Z80 PIOを用いたプリンタ・インターフェース



使うときには参考になります。

● PIAのもつ機能

PIAの機能を列挙してみると、

- ▶ たんなるたれ流し式のラッチI/O
 - ▶ ストロープ付きのI/O
 - ▶ ハンドシェイクによる転送
 - ▶ 上記のデータ転送をプログラミングで操作可能
- などでしょう。ですから並列入出力のほとんどの応用はPIAで実現できるわけです。

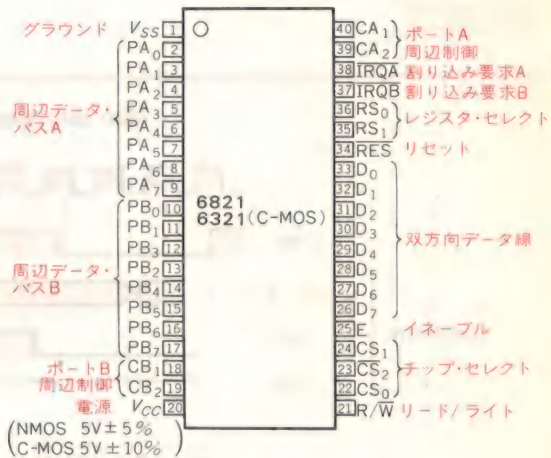
● PIAの周辺インターフェース

図18にPIAの端子配置を示します。PIAは外部デバイスとのインターフェースに使われるLSIですから、当然、周辺インターフェース用の端子があるわけです。これらは、8ビット単位でA、Bの2チャンネル分あり、データ用のPA₀~PA₇, PB₀~PB₇, ストローブ信号やインタラプトのトリガ入力などに使われるCA₁, CA₂, CB₁, CB₂があります。

◆ MPUがアクセスできるPIAの内部レジスタ

さて、PIAの内部構成をのぞいてみましょう。図19にPIAの内部ブロック図を示します。MPUがアクセス可能なPIAのレジスタは、

〈図18〉PIAの端子配置



- ▶ データ方向指定レジスタ
- ▶ 周辺インターフェース・レジスタ
- ▶ コントロール・レジスタ

で、AポートとBポートあわせて全部で6本あります。

● DDRA(B) (データ方向指定レジスタ)

データ方向指定レジスタは、I/Oデータの方角を決


```

; program Z80502.MAC
; Z80-PIO test routine
; 85/02/10 Y.Kanzaki
;
001D      Z80 PIOの各ポートのアドレス
001C      piopac equ 01Dh ; PIO port address
001F      piopad equ 01Ch ;
001E      piopbc equ 01Fh ; PIO port address
001E      piopbd equ 01Eh ;
000F      Z80 PIOモード0の設定コマンド
00CF      cmd_out equ 00Fh ; PIO
00CF      cmd_bit equ 0CFh ; PIO
00F0      cmd_io equ 0F0h ; in D7-D4 out D3-D0
;
0000'      org
0002'      3E 0F Z80 PIOのモード pioint: ld A,cmd_out ; ポートAをモード0に設定
0004'      D3 1D 設定ルーチン out (piopac),A
0006'      3E CF ld A,cmd_bit ; ポートBをビット制御モードにして、各ビットの入出力を設定
0008'      D3 1F out (piopbc),A
000A'      3E F0 ld A,cmd_io
000C'      D3 1F out (piopbd),A
000C'      C9 ret
;
000D'      DB 1E プリンタの状況の lstat: in A,(piopbd) ; プリンタが受信可能でなければA=0FFHとして
000F'      CB 67 チェック bit 4,A ; もどる。
0011'      3E FF ld A,0FFH ; プリンタが受信可能なら
0013'      C8 ret Z ; A=0
0014'      AF xor A
0015'      C9 ret
;
0016'      CD 000D' STBはZ80 PIOより出力される list: call lstat ; プリンタが受信可能になるまでループする
0017'      B7 or A
001A'      2B FA jr z,list ;
001C'      79 ld A,C ; Cレジスタにセットされた
001D'      D3 1C out (piopad),A ; データを、プリンタに出力
001F'      C9 ret ; する
;
end

```

定するレジスタで、各ポート、各ビットごとに指定できます。図20のようにAポートのビット3, 4, 7を出力に、そのほかは入力に割り付けるといような使い方ができるわけです。

なぜ、DDRA(B)が最初に登場するのかというと、リセット直後にPIAの内部アドレス0または内部アドレス2を読み込むと、このレジスタの内容が得られます。また、書き込み動作を行っても同様にDDRA(B)に書き込まれます。すなわち、PIAの最も基本的な項目であるデータ方向は、初期化のときに決まっていなければならないからです。

● PRA(B) (周辺インターフェース・レジスタ)

周辺デバイスとMPUのデータ・インターフェースを行うPRA(B)とDDRA(B)は内部アドレスが同じですが、コントロール・レジスタCRA(B)のビット2の値によって切り替えることができます。すなわち、CRA(B)のビット2が0のとき、内部アドレス0, 2はDDRA(B)が選択されていますが、ここを“1”にしてやるとPRA(B)が現れるというわけです。図21にPIAレジスタ選択の概念図を示します。

このレジスタは、I/Oデータそのものを取り扱うもので、MPUがPRA(B)をリード/ライトすることが、すなわち周辺デバイスとのデータのやりとりになります。

```

( Z80-PIO test program )
( 1985/02/10 code Y.Kanzaki )
program piotest;
const
    piopac = $1D;
    piopad = $1C;
    piopbc = $1F;
    piopbd = $1E;
    cmd_out = $0F;
    cmd_bit = $CF;
    cmd_io = $F0;
    Z80 PIOの各ポート・アドレスを定数として定義する
    モード制御コマンドを定数として定義する

procedure pioint;
begin
    port[ piopac ] := cmd_out;
    port[ piopbc ] := cmd_bit;
    port[ piopbd ] := cmd_io;
    Z80 PIOのモード設定を行って
end;

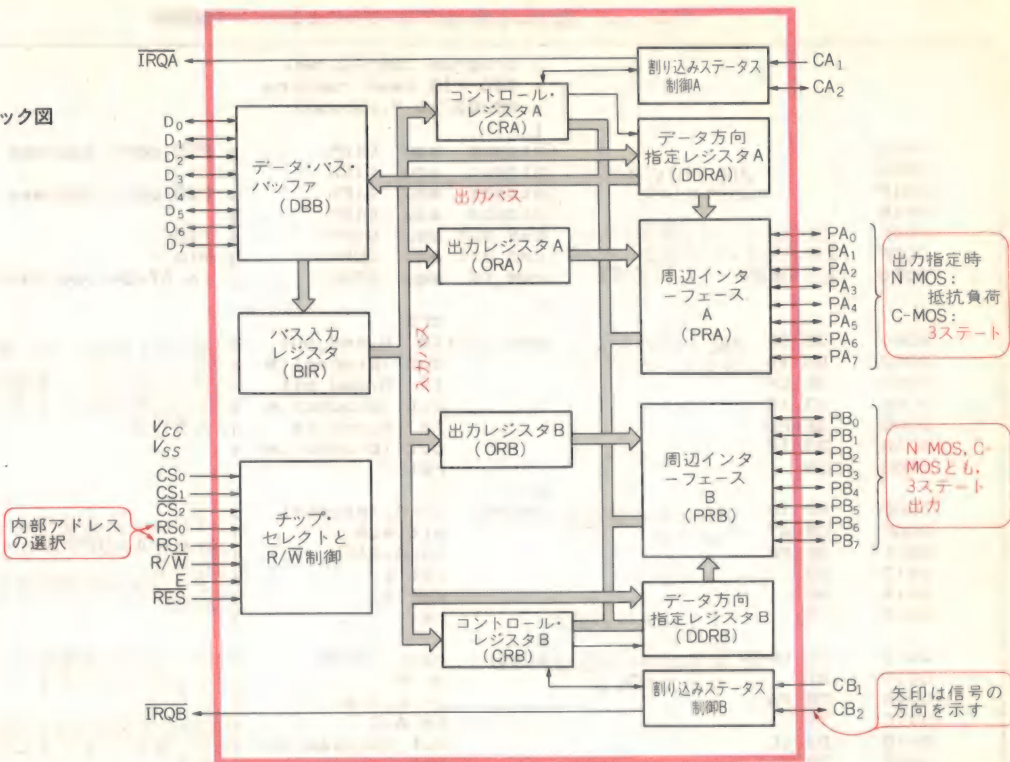
function lstat:boolean;
begin
    lstat := (port[ piopbd ] and $10) = 0;
    プリンタの状態チェックの関数
end;

procedure list( data : byte );
begin
    repeat
        until lstat;
        port[ piopad ] := data;
        プリンタが受信可能になるまでI/Oポートpiopadへ出力データを書き出す
    end;

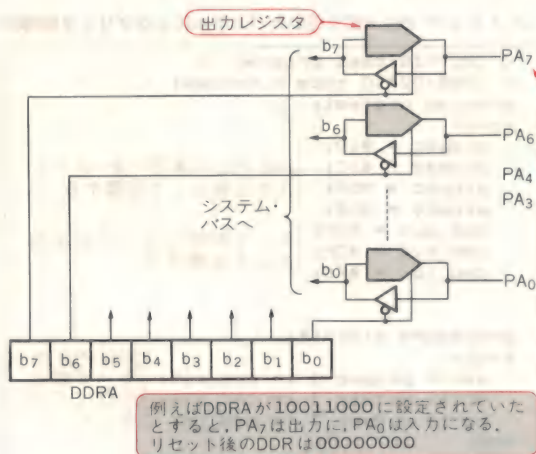
begin
    pioint;
    list( $31 );
    list( $0D );
    テストのために一つのCRを出力するプログラム
end.

```

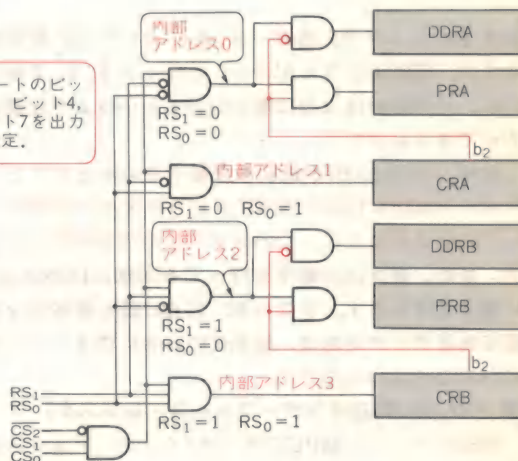
〈図19〉
PIAの内部ブロック図



〈図20〉DDR(データ方向指定レジスタ)の概念



〈図21〉PIAのレジスタの選択



● CRA(B) (コントロール・レジスタ)

MPUと周辺デバイスとのデータ転送をする場合、両者間の処理能力の関係や突発的なデータの送信により、正しいデータを見逃す可能性があります。したがって、PRA(B)のたんなるリード/ライトでは信頼性のあるデータの転送はできません。

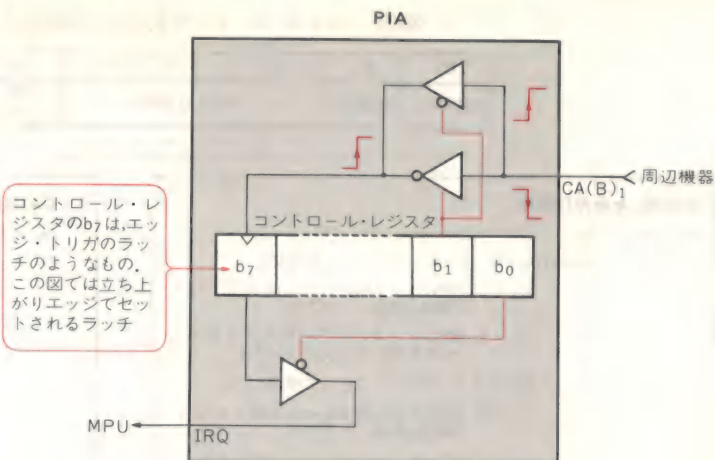
ですから、ストロブやハンドシェイクという手法を使うことになります。CA(B)_{1,2}端子はこのために使われるもので周辺制御線とも呼ばれます。コントロール・レジスタの役割はこれらの周辺制御線をコント

ロールすることと、前述したDDRA(B)とPRA(B)の切り替えおよび割り込み信号の処理です。

PIAが難解といわれるのは、この周辺制御線をどのようにうまく目的のデータ転送に応用するかといった点でしょう。そのためにCA(B)_{1,2}について詳しく考察してみます。

CA(B)₁は入力専用で、外部事象の変化を捕らえることができます。CA(B)₂は入出力どちらにも設定でき、入力に設定した場合にはCA(B)₁と同じような動作をします。これらの設定は、もちろんコントロー

〈図22〉
CA(B)₁関連のコントロール・レジスタ



ル・レジスタによって行います。

◆ コントロール・レジスタの働き

図22にCA(B)₁ 関連のコントロール・レジスタの働きを示します。関連ビットの意味は、

- ▶ ビット 7 は、CA(B)₁ の変化(ビット 1 で設定)があったことを示すフラグで読み出し専用です。
- ▶ ビット 0 は、ビット 7 の内容を割り込み信号として、

ハード的にMPUへ出力することを許すフラグです。すなわち、CA(B)₁に変化があったときのインタラプト・イネーブル/ディセーブルを決めます。

- ▶ ビット 1 はCA(B)₁の変化の方向を決めるもので、周辺デバイスのストローブ信号を↑エッジでも↓エッジでも捕らえることができますから、周辺デバイスの仕様に合わせることができます。
- なお、CA(B)₂を入力にセットするときは、ビット

現場技術者実戦シリーズ

好評発売中

改訂 高周波回路設計ノウハウ

——部品/回路/実装のポイント徹底説明——

吉田 武 著 2色刷 A5判 304頁 定価2,957円(税込)

本書は、1985年に初版を発行して以来、今日までロングセラーを博してきた「高周波回路設計ノウハウ」をもとに、最新の技術動向を考慮し、大幅に改訂を加え再デビューしたものです。内容は次のとおり、広範囲にわたっているため、高周波回路に従事するエンジニアには座右の書として、十分活用できるものと思われます。

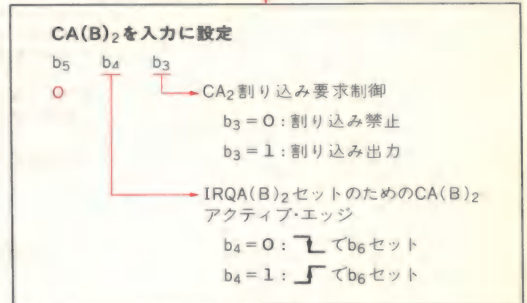
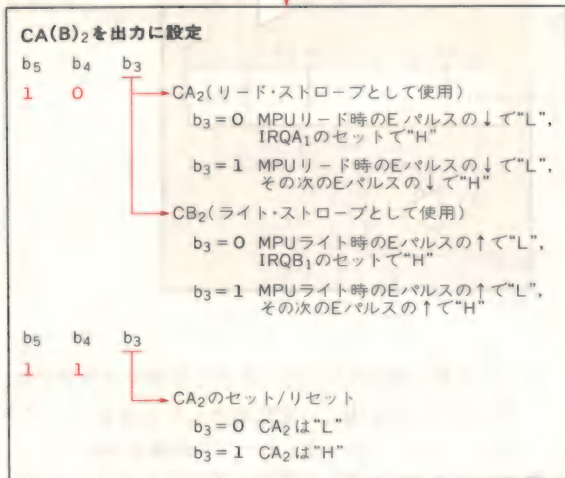
〈内容〉

第1章；高周波部品の知識と実装のノウハウ、第2章；高周波回路の実験・試作のノウハウ、第3章；高周波増幅回路、第4章；高周波発振回路、第5章；フィルタ/トラップ回路、第6章；各種高周波回路。



〈図23〉コントロール・レジスタのビット割り付けとCA(B)端子の設定

b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀
IRQA(B) ₁	IRQA(B) ₂	CA(B) ₂ 制御			DDR アクセス	CA(B) ₁ 制御	



5を“0”にします。こうすることにより、CA(B)₂もCA(B)₁と同じ機能をもつことになります。このとき、ビット3がビット0と、ビット4がビット1と同様の意味をもちます。

● CA₂とCB₂は違う

図23にCA(B)の制御のしかたをまとめて示します。ここまでの解説とPIAの内部ブロック図からは、CA₂とCB₂の違いはわかりません。しかし、この二つの信号線には、大きな違いがあるのです。図23をみてください。CA₂、CB₂を出力に設定したとき、

▶CA₂はAポートからデータを読み出すことによって

セットされる。

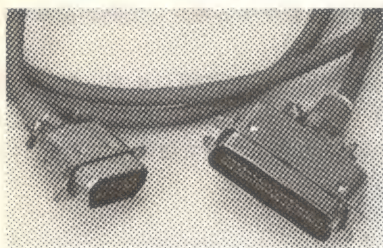
▶CB₂はBポートにデータを書き込むことによってセットされる。

ということになります。この意味を考えると、周辺デバイスに対して、

▶CA₂が「データを受け取った」

▶CB₂が「データを送った」

の確認信号を出していると取れるわけです。したがって、PIAをハンドシェイクで使う場合、Aポートは入力に、Bポートは出力に使うと便利であることがわかります。



§ 1-3

セントロニクス↔RS-232C 変換器の製作

土屋 哲/矢吹貞人

セントロニクス・インターフェースとRS-232Cインターフェースは、パソコンの標準インターフェースといってもよいくらい、ほとんどのパソコンで最低どちらか一種が標準装備されています。

また、パソコンと接続する周辺装置も、セントロニクス・インターフェース、RS-232Cインターフェースを備えたものが多くなっています。

そこで、セントロニクス→RS-232C変換器やRS-232C→セントロニクス変換器があれば、パソコンと周辺装置およびほかのパソコンとの接続の可能性が増大することになり、パソコン間でプログラムやデータの転送を行ったり、機器の有効利用が図れると考え、セントロニクス↔RS-232C変換器を製作しました。

回路構成

本器のブロック図を図1に示します。本器はUART(Universal Asynchronous Receiver Transmitter)を中心にして、大きく三つの部分に分けられます。

一つは、ボーレート・ジェネレータです。UARTの下側の部分で、UARTに必要なボーレートに対応したクロックを供給します。

もう一つは、RS-232C→セントロニクス変換部です。UARTおよびUARTの左側の部分で、RS-232Cのシリアル・データをセントロニクスのパラレル・データに変換します。

残りの一つはセントロニクス→RS-232C変換部です。UARTおよびUARTの右側の部分で、セントロニクスのパラレル・データを、RS-232Cのシリアル・データに変換します。

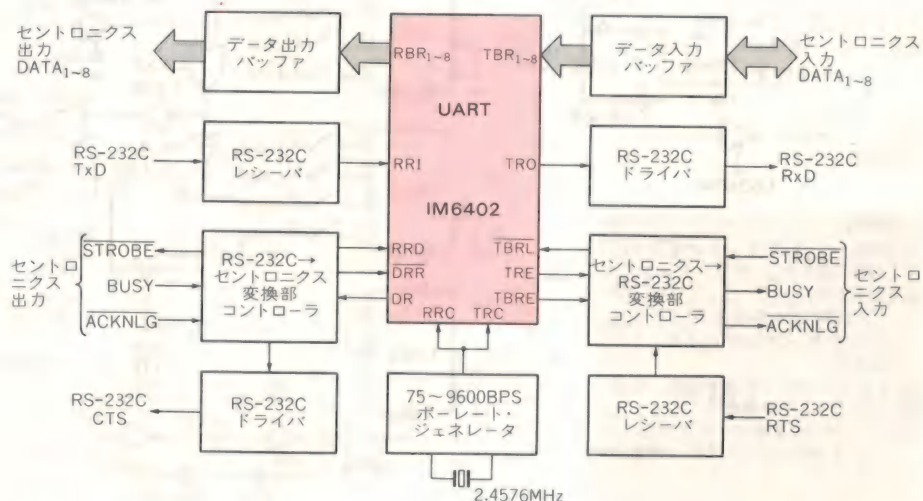
本器の回路図を図2に示します。次に各部分について説明します。

● ボーレート・ジェネレータ

ボーレート・ジェネレータ部は、2.4576MHzの水晶とLS04による発振回路の出力を、LS93を3個用いて分周する回路です。ボーレート 75, 150, 300, 600, 1200, 2400, 4800, 9600のそれぞれ16倍のクロック(UART, インターシル社のIM6402は、ボーレートの16倍のクロックを必要とする)として取り出しています。

本器のクロックは、受信部、送信部とも同一の周波数を用いています。受信部、送信部別の周波数(すなわち別のボーレート)で動作させることも可能です。ボーレートの切り替えは、SW₁(ボード上のDIP SW)により行います。

〈図1〉
セントロニクス↔
RS-232C変換器の
ブロック図



RS-232C→セントロニクス変換部は、図1,図2のUARTおよびUARTの左側の部分です。

トップ・ピット長の設定がそれぞれのピンの電圧レベルを設定するだけで行えます。本器のようにハードウェアのみで変換器を製作する場合に回路が簡単になり有利です。

次に回路の動作について説明します。なお**本器のRS-232Cコネクタは、モデム定義**となっていますので、各信号の名前、役割について注意してください。

SW₂

- ① (ON)パリティ・チェックを行う,
(OFF)パリティ・チェックせず
- ② (ON)1ストップ・ビット(OFF)データ長5ビットの場合は1.5ストップ・ビット, それ以外は2ストップ・ビット
- ③ } データのビット長を指定, 設定については右表参照
- ④ }
- ⑤ ①が(ON)のときパリティ指定,
(ON)奇数パリティ, (OFF)偶数パリティ

ワード長 (ビット)	SW ₂ ③	SW ₂ ④
5	(ON)	(ON)
6	(ON)	(OFF)
7	(OFF)	(OFF)
8	(OFF)	(OFF)

① まず、RS-232Cの送信データ(TxD)はRS-232CレシーバMC1489LでTTLレベルに変換され、UARTの受信部のRRI端子に加えられます。

② 次に、UARTの受信部はRRI端子に加えられたシリアル・データのスタート、データ、パリティとストップ・ビットを受け、パリティとストップ・ビットが正しいことを確認した後、パラレル・データに変換します。

③ このとき、UARTのDR端子は“H”レベルに変わりますので、この信号を反転し、UARTのRRD端子に加え、UARTのパラレル・データ(RBR₁～RBR₈)を読み出し、LS367Aによるバッファを通してセントロニクス出力のDATA₁～DATA₈とします。

④ 一方、DRが“L”から“H”に変わったときから約1 μ s遅れた約1 μ s幅のパルスが、LS123による単安定マルチバイブレータ2段の回路により作られ、セントロニクス出力のSTROBEとなります。

⑤ セントロニクス側の動作が完了し、セントロニクス出力のACKNLGが“H”から“L”に変わったとき、LS123の単安定マルチバイブレータが約0.5 μ sのパルスを出します。そして、これをUARTのDRRに加えDRを“L”レベルにし、UARTの受信部が次のシリアル・データを受信できるようにします。

⑥ 一方、DRとセントロニクス出力のBUSY信号のORをとり、RS-232CドライバSN75150Pを通してRS-232Cの送信可(CTS)に出しています。

このRS-232C→セントロニクス変換部のタイムチャートを図3に示します。本器では、セントロニクス関係のタイミングは、エプソン社のプリンタTP80のタイミングを参考にして設計してあります。

シリアル信号のデータ長、パリティ・チェックの有無、ストップ・ビット長の設定は、セントロニクス→RS-232C変換部と共通で、SW₂(ボード上のDIP SW)で行います。SWの設定法については、図2を参照してください。

RS-232Cコネクタは、セントロニクス→RS-232C変換部と共通です。RS-232Cコネクタの信号配置を表1に示します。本変換部では、動作に必要な信号は送信データ(TxD)、送信可(CTS)および信号接地(SG)のみですが、相手方の動作のためにデータ・セット・レディ(DSR)、キャリア検出(CD)の各々は、+12Vにプルアップしてあります。

セントロニクス出力コネクタの信号配置を表2に示します。

● セントロニクス→RS-232C変換部

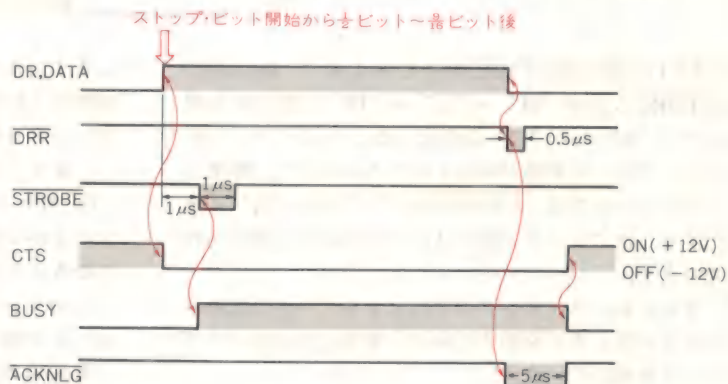
セントロニクス→RS-232C変換部は、図1、図2のUARTおよびUARTの右側の部分です。

次に、回路の動作について説明します。

① まず、セントロニクス入力₁のDATA₁～DATA₈をバッファLS541を通して、UARTの送信部のTBR₁～TBR₈に加えます。

② 次に、セントロニクス入力₁のSTROBEをパルス幅約0.5 μ sの単安定マルチバイブレータで整形して、

〈図3〉⁽¹¹⁾
RS-232C→セントロニクス変換部の
タイムチャート



〈表1〉
RS-232Cコネクタの
信号配置

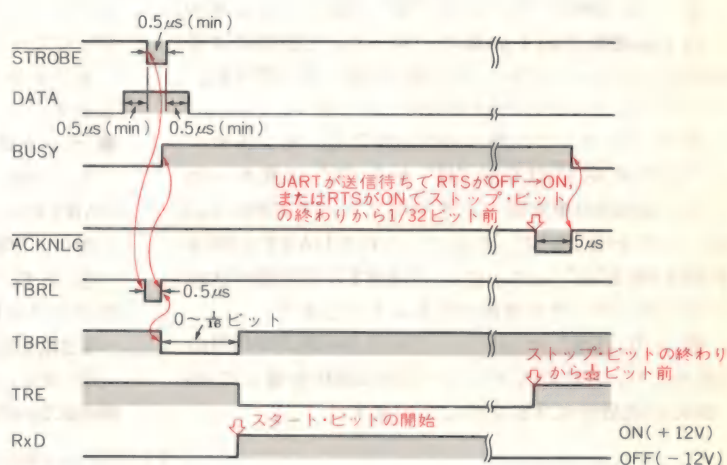
ピンNo.	信号名	方向	機 能
1	FG		保安アース
2	TxD	→本器	送信データ
3	RxD	←本器	受信データ
4	RTS	→本器	送信要求(本器ではセントロニクス→RS-232C変換部のハンドシェイクに使用)
5	CTS	←本器	送信可(本器ではRS-232C→セントロニクス変換部のハンドシェイクに使用)
6	DSR	←本器	データ・セット・レディ(本器では+12Vにプルアップ)
7	SG		信号アース
8	CD	←本器	キャリア検出(本器では+12Vにプルアップ)

(注) 本器の信号配置はモデム定義になっている

〈表2〉
セントロニクス出力コネクタの信号配置

信号 ピンNo.	リターン側 ピンNo.	信号名	方向	解 説
1	19	STROBE	出	定常“H”. パルス幅約1 μ s
2	20	DATA ₁	出	8ビットのデータ信号線. ストローブ信号の“L”パルスの前約1 μ s からACKNLGが“L”になるまでデータが 確定している
3	21	DATA ₂	出	
4	22	DATA ₃	出	
5	23	DATA ₄	出	
6	24	DATA ₅	出	
7	25	DATA ₆	出	
8	26	DATA ₇	出	
9	27	DATA ₈	出	
10	28	ACKNLG	入	周辺装置からのデータ入力の完了を示す信号
11	29	BUSY	入	周辺装置が入力の可否を示す
16		0 V		ロジックGNDレベル
17		CHASSIS- GND		筐体グラウンド
33		GND		

〈図4〉
セントロニクス→RS-232C変換部の
タイムチャート



UARTの送信部のTBRL端子に加えます。UARTではTBRL入力が“H”→“L”→“H”と変化する間のうち“L”から“H”に変化するときにパラレルからシリアルへの変換が開始されるのに対して、標準セントロニクスでは、STROBEが“H”から“L”に変化するときにデータの読み込みが行われると規定されています。

そのため、タイミングのずれによるデータの読み込みミスが生じるのを防ぐために、単安定マルチバイブレータを用いています。

③ このTBRLの立ち上がりにより、セントロニクスのパラレル・データがシリアル・データとなり、UARTのTRO端子に出力され、RS-232CドライバSN75150Pを通してRS-232Cの受信データ(RxD)に出力されます。

④ UARTのビジー状態は、UARTのTREとTBREのNANDで表されます。RS-232Cの送信要求(RTS)がONの場合は、このビジー状態を表す信号の立ち下がりパルス幅約5 μ sの単安定マルチバイブレータを働かせ、セントロニクス入力のACKNLGと

します。また、UARTが送信待ちの場合は、RTSがOFFからONに変わったときにこの単安定マルチバイブレータを働かせ、セントロニクス入力のACKNLGとします。

⑤ セントロニクス入力のBUSY信号は、TBRLの立ち上がりからACKNLGの立ち上がりまでの間“H”になるようになっています。

このセントロニクス→RS-232C変換部のタイムチャートを図4に示します。

本変換部では、動作に必要なRS-232Cの信号は、受信データ(RxD)、送信要求(RTS)および信号接地(SG)のみです。

RTSによる制御法は、ソード社のM223のモデム定義のRS-232Cコネクタを参考にしました。

セントロニクス入力コネクタの信号配置を表3に示します。

◆ 使用例

本器はセントロニクス、RS-232C両入力対応のPTPインターフェースの一部として製作されたも

〈表3〉
セントロニクス入力コネクタの信号配置

信号 ピンNo.	リターン側 ピンNo.	信号名	方向	解 説
1	19	$\overline{\text{STROBE}}$	入	定常“H”、“L”から約0.5 μs 後にデータ読み込み、パルス幅0.5 μs 以上
2	20	DATA ₁	入	8ビットのデータ信号線。 ストロブ信号の“L”パルスの前後0.5 μs 間はデータが確定していなければならない
3	21	DATA ₂	入	
4	22	DATA ₃	入	
5	23	DATA ₄	入	
6	24	DATA ₅	入	
7	25	DATA ₆	入	
8	26	DATA ₇	入	
9	27	DATA ₈	入	
10	28	$\overline{\text{ACKNLG}}$	出	データ入力の完了を示す、割り込み用5 μs
11	29	BUSY	出	本器が次のデータの入力の可否を示す、“L”で可
12		PE	出	“H”で用紙なし、“L”に設定してある
16		0 V		ロジックGNDレベル
17		CHASSIS-GND		筐体グラウンド
32		ERROR	出	“H”に設定してある
33		GND		

ので、RS-232C→セントロニクス変換部は、ソードM223とセントロニクス入力PTPあるいは、エプソン社のプリンタTP80〔またはFP100(現在のVPシリーズの前がMPシリーズで、その前のシリーズにあたる)〕との間に入れて、ボーレート9600で良好に動作中です。

また、エプソン社のハンドヘルド・パソコンのHC20とプリンタFP80との間に入れて、ボーレート4800で良好に動作しました。

セントロニクス→RS-232C変換部は、データ・ジェネラル社のミニコンNOVA 3のライン・プリンタ出力(セントロニクス入力のプリンタと接続できるように信号の論理を変更してある)とソード社のM23との間に入れて、NOVA 3上で開発したプログラムやデータ・ファイルをM23に移植するのに用いました。

移植に際しては、ソフトウェア開発の負担はまったくといってよいほどなく、NOVA 3側では、プログラムやデータをライン・プリンタに印字する命令を与えるのみです。M23側では、エディタでRS-232C入力(ボーレート1200に設定、RTS信号による制御は有効でない)を入力ファイルに指定し、データを入力し、フロッピー・ディスク上の出力ファイルに書くだけです。

また、ソードM23のセントロニクス・プリンタ出力とエプソンHC20との間に入れて、BASICのプログラムを転送してみました。ボーレート300まで良好に動作しましたが、ボーレート600以上でHC20側のパッ

ファ・オーバフロー・エラーが起り、動作しませんでした。これは、RS-232Cの制御信号RTSによるハンドシェイクが行われなかったために、変換器からのRS-232C出力がたれ流し状態になっているためです。

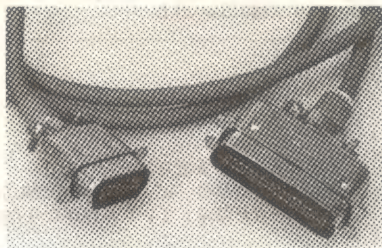
◆ 使用上の注意点

本器のようにセントロニクス→RS-232C変換する場合、両インターフェースの動作に起因する問題点があります。それは、セントロニクスがハンドシェイクで動作するのに対応して、RS-232Cもハンドシェイクで動作する必要があるのに、**現実にはRS-232Cは必ずしも制御信号線を用いたハンドシェイクを行って使われているとは限らない**ことにあります。

このため、上で述べた使用例以外では、原理的に本器が使用できない場合があることも考えられます。また、原理的には可能でも信号線の接続法の違いにより動作しない場合もありますので、個々のパソコンおよび周辺装置のタイミング、信号線の役割などについて十分検討の上使用してください。

とくに、RS-232Cの場合、使用する信号線については機器によってまちまちで、その信号線の役割も様々に使われています。十分注意してください。

なお、RS-232Cのハンドシェイクの問題は、ソフトウェアでRTSやDTRなどの制御信号をコントロールすることにより解決できる場合もあります。



簡易型プリンタ・バッファの製作

斉藤洋司/生沼守英

最近の半導体技術の進歩は著しく、とくにICメモリの高密度化が進んでいます。そのおかげで、パソコンの記憶容量は飛躍的に増大しました。しかし、プリンタなどの出力装置の高速化はあまり望めず、ユーザーの要求に十分対応しきれていないのが現状です。

我々の研究室においても、測定データをプリンタやプロッタへ出力する際、それらの動作速度の遅さが気になっていました。この問題を解決し得るのが、並列処理により高速化を行うプリンタ・バッファ⁽¹³⁾です。ただし、我々の用途ではそれほど大容量のバッファは必要でなかったことや、市販品はプリンタ切り替え機を兼ねていたりして、適当な価格の製品が見当たらなかったため、製作を試みることにしました。

比較的製作経験の浅い人でも製作できるように、再現性を重視しながら可能な限り回路を簡素化しました。ただし、少なくともテストとオシロスコープは、デバッグのために必要になるでしょう。

なお、本機はプリンタ切り替え機を兼ねたプリンタ・バッファであり、セントロニクス⁽¹⁴⁾準拠のインターフェースを備えたパソコンやプリンタ、プロッタなどに使用できます。

本機の回路構成

図1に本機の全回路図を示します。Z80 CPUを中心としてRAM、ROM、PPIなどにより構成しています。次に、それらの部品の選択および使い方について簡単に説明します。

● メモリ

まずメモリですが、RAMには再現性を高めるためスタティックRAM(SRAM)を用いることにしました。その場合、必要に応じて56Kバイトまでの範囲で、8Kバイトごとにバッファ容量を増減することができます。

回路構成を変えずに、さらにコストを抑えたい場合は、疑似スタティックRAM(PSRAM)⁽¹⁵⁾を用いるとよいでしょう。PSRAMは、ピン配置がSRAMと同じで、使い方も同様にできますが、内部はダイナミッ

クRAMですので、リフレッシュのため若干の回路変更を要します。

256KビットPSRAM(HM65256：日立)の場合、パッケージのピン数の制限から、64KビットPSRAMにはあるRFSH端子のような端子がありません。そのため、図2に示すタイミングでOEおよびCS端子にパルスを加えることにより、自動的にリフレッシュが行われるようになっています(これをオート・リフレッシュ・モードと呼ぶ)。

本機では、図1の赤枠内のRAM₁~RAM₄の代わりに、図3の回路を用いればよいのですが、LS00とLS08を追加する必要があります。

ROMには、2764または27128を用いました。プログラム・サイズは500バイト以下ですので、ほんの一部しか使用していないことになります。

● PPI 8255A

次にPPIについてですが、8255Aをデータ入出力のポートとして用いています。各ポートの割り当ては、表1を参照してください。

データ入力ポートのみ、モード1⁽¹⁶⁾で使用され、パソコンからSTB信号が入ると、ハード的にBUSYおよび割り込み要求信号が発生するようになっています。出力ポートはすべてモード0で、STBの制御はビットのセット/リセットにより行います。なお、ACKは無視しています。

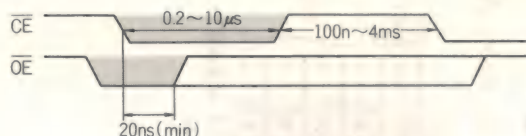
セントロニクス仕様では、LS14相当のシュミット・トリガを抵抗でプルアップした回路を入力とし、7406などのオープン・コレクタで出力するのが原則となっています。

しかし、とくにノイズ環境が悪くない限り、図1のように8255Aで直接入力を行っても問題ありません。

出力ドライバの7407も不要な場合が多いのですが、8255Aのシンク能力が2.5mAと小さいため、プリンタなどの入力が2kΩ以下でプルアップされている場合は、このドライバを省略すると動作の保証はできません。

例えば、エプソン社のTP、MPシリーズのプリンタでは、プルアップ抵抗は3kΩですので、8255Aによ

〈図2〉⁽¹⁴⁾ 256K PSRAMのリフレッシュ・タイミング



るダイレクト・ドライブが十分可能ですが、日本電気のPCシリーズのプリンタでは1 k Ω ⁽¹⁷⁾ですので、直接接続するのは避けたほうが無難です。

● コネクタの接続

パソコン本体やプリンタなどへのコネクタの接続は、必ずそれぞれのハードウェア・マニュアルをよく見て行ってください。とくに、パソコンによっては、SELECT、FAULT、PAPEREND(PE)などの入力端子をもつ機種があり、**それらの端子をプルアップ、またはGNDに接続しないとプリンタ・エラーとなり、動作しない場合があります。**

また、本機とパソコンなどとの接続には、少なくとも11芯のケーブルが必要です。シールド付きのツイステッド・ペア構造多芯ケーブルが最良ですが、通常のフラット・ケーブルでも十分です。ただし、ケーブルの長さは必要以上に長くしないようにしてください。

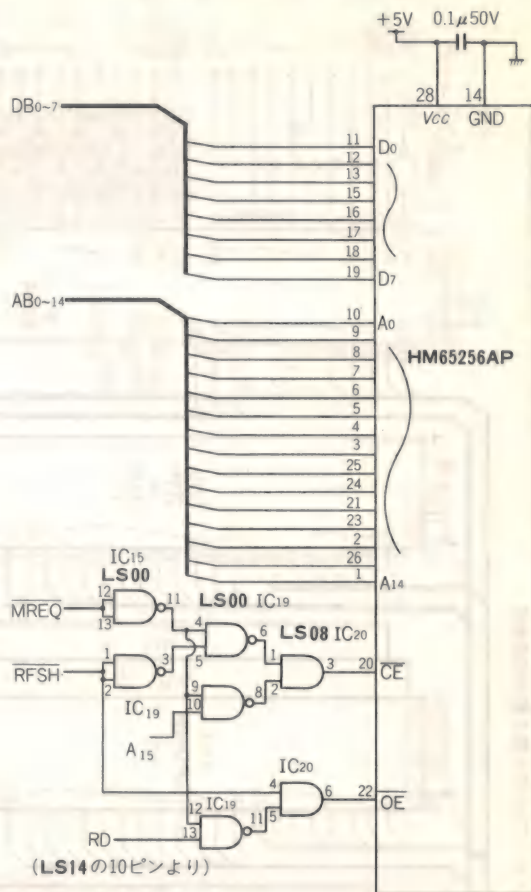
本機に使用するコネクタの選択は、入力にはほとんどのプリンタに採用されている36ピンのアンフェノール・コネクタを使用し、出力にはパソコン本体のプリンタ出力に用いられているものと同じコネクタを使用すれば、ケーブルの互換性が保てるので便利です(図4)。

● CPUのクロック

次に、CPUに供給するクロックについて述べます。クロックは、図1では、TTLゲートとCRで発生させていますが、コンデンサをクリスタルに置き換えても結構です。ただし、クロック周波数は、おおむね2~3.5MHzの範囲に選んでください。

その理由は、パソコンから本機へのデータ転送速度は、主にパソコンの出力速度で決まってしまうので、クロック周波数を上げてほとんど効果がないためで

〈図3〉 PSRAM使用時の回路変更



す。さらに、クロック周波数を4 MHzにした場合、クロック・パルスのデューティ比が1:1でなくなつてZ80Aの規格を満足しなかったり、ROMの品種により動作しない恐れ⁽¹⁸⁾があります。

● 電源について

次に、ボードに供給する電源ですが、+5 V、250~300mAが必要になります。本機の場合は、トランスで降圧し、3端子レギュレータで安定化した電源を用いました。7805は発熱しますので、必ず放熱板を

〈表1〉
PPI(8255A)の使い方

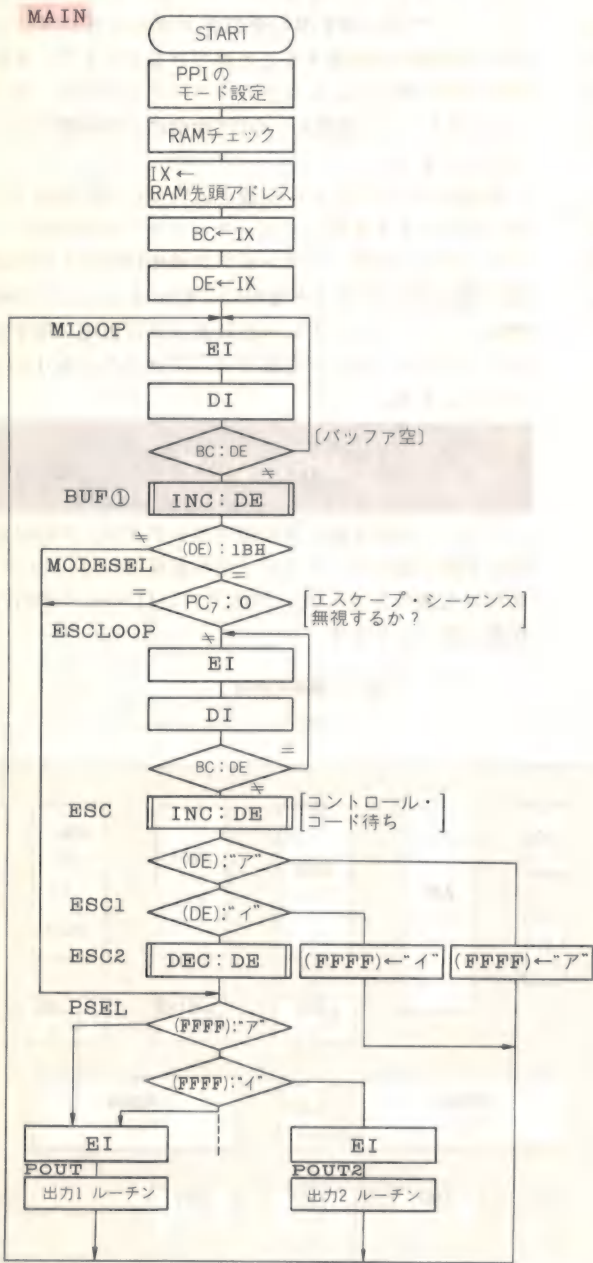
	PPI ₁ (IC ₁₀)	PPI ₂ (IC ₁₁)
アドレス	00H ポートA (入力, モード1) 01H ポートB (出力, モード0) 02H ポートC (入力, 出力) 03H 制御	04H ポートA (出力, モード0) 05H ポートB (出力, モード0) 06H ポートC (上位入力, 下位出力) 07H 制御
ポートC割当て	ビット3 (出) ACK } 4 (入) STB } パソコンへ 5 (出) BUSY } 0 (出) STB } 出力1 6 (入) BUSY } 7 (入) モード切り替え 1 (出) 未使用 2 (出) 未使用	ビット0 (出) STB } 出力2 6 (入) BUSY } 1 (出) STB } 出力3 7 (入) BUSY } 2 (出) 未使用 3 (出) 未使用 4 (入) 未使用 5 (入) 未使用

つけてください。

ソフトウェアについて

プログラムの流れ図を図5に示します。基本的な働きは、入力ポートにSTBパルスが入って、ハード的に割り込み要求が起これば、割り込み禁止状態となり、かつバッファ・メモリがあふれていなければ、データを1文字入力し、メモリに格納することと、バッファ

〈図5〉プログラムのフローチャート

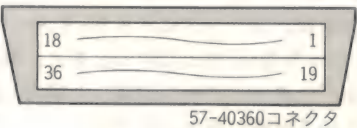


内のデータが空でなければデータを出力すること、および出力ポートの切り替え命令を解釈することです。

割り込みには、Z80モード1を用いました。割り込みルーチンに入ると、条件を満たせばデータをPPI₁(IC₁₀)のポートAより入力し、レジスタBCの表す番地のメモリに格納します。

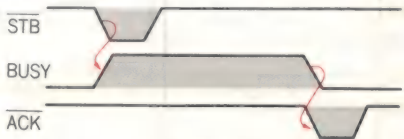
出力は、条件を満たせばF F F F H番地の内容が示す出力ポートに、レジスタDEの表すメモリの内容を出力するようになっています。

〈図4〉⁽¹⁾ セントロニクス・インターフェースのコネクタ・ピン配置とタイミング



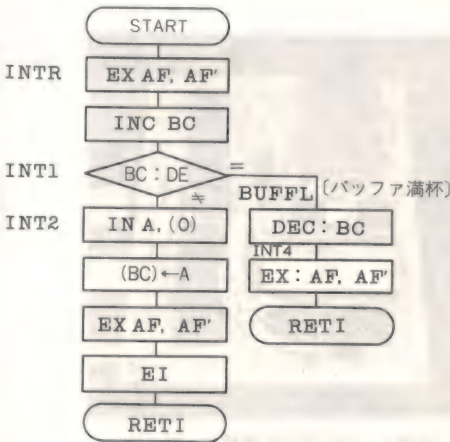
1	STB	19	GND
2	DATA 1	20	
3	DATA 2	21	
4	DATA 3	22	
5	DATA 4	23	
6	DATA 5	24	
7	DATA 6	25	
8	DATA 7	26	
9	ACK	27	
10	BUSY (READY)	28	
11	(PE)	29	
12	(SELECT)	30	
13	(OV)	31	
14	NC	32	
15	OV	33	
16	シャーシGND (+5V)	34	
17		35	
18		36	

(a) コネクタ接続(日本電気製のプリンタの場合)



(b) ハンドシェイク・タイミング

割り込みルーチン



● 出力ポートの切り替え

出力ポートの切り替え(F F F F H番地にデータを書き込む動作)は、エスケープ・シーケンスにより行います。

具体的には、1 B Hに続くデータがB 1 Hならば出力1が選択され、B 2 Hならば出力2が選択されます。これらのキャラクタ・コードは、カタカナの“ア”または“イ”に相当し、プリンタでは機能コードとして使用されることはまずありません。また、このエスケープ・シーケンスは、現在の国産のパソコンでは必ずサポートしていると考えられます。

ただし、このエスケープ・シーケンスによる切り替え方法の欠点は、プリンタにドット・イメージ(画面のハード・コピーを含む)を出力するときに、偶然先に述べたようなデータ・パターンが現れた場合、誤動作してしまうという問題があることです。

この問題をすべての機種種のプリンタに対し、ソフトで解決することはほぼ不可能です。そこで、考えられる解決法としては、パソコンのI/Oポートを用いて切り替えを行う方法⁽¹⁾、スイッチで切り替える方法な

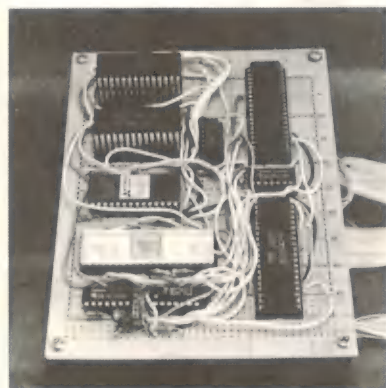
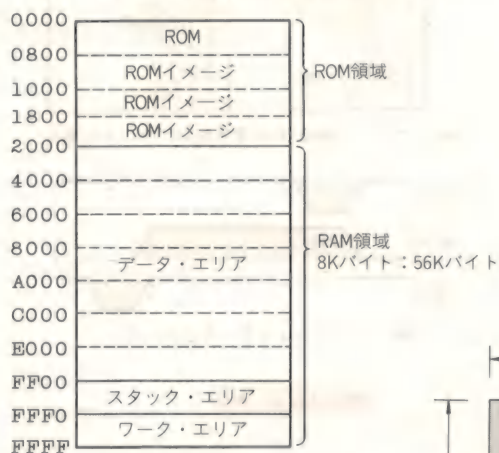
どがあります。

本機では、その両方の方法が使えるような回路とし、今回はとりあえずスイッチで、通常モードと切り替えコードを無視するモードとに切り替えられるようにしました。実際には、ドット・イメージを出力するときは、IC₁₀の10ピンがGNDに接続されるようにSW₂を閉じてください。

パソコンのI/Oポートで、モード切り替えを行う方法は、その端子にSW₂の代わりにリレー接点を接続するか、0 Vまたは5 Vの信号を加えることにより行います。パソコンにカセット・インターフェースがあり、モータON/OFF用の接点がサポートされていれば、その接点を利用することができでしょう。それができない場合は、かなりのコストアップになりますが、I/Oカードを増設し、OUT命令により制御することになります。

ROMへのプログラムの書き込みは、CP/M80上でワード・マスタを用いてソース・ファイルを作成し、アセンブルした後、トランジスタ技術1985年1月号掲載の簡易型ROMライタを用いて行いました。その場合は、バッファ・アドレスを0 8 0 0 H番地に指定します。メモリ・マップを図6に、プログラムをリスト1に示します。

〈図6〉プログラムのメモリ・マップ

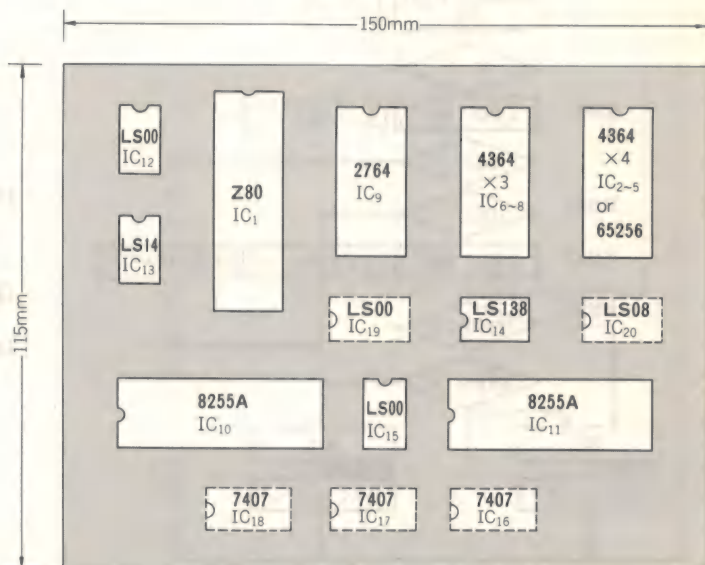


〈写真1〉56K RAMを実装したボード

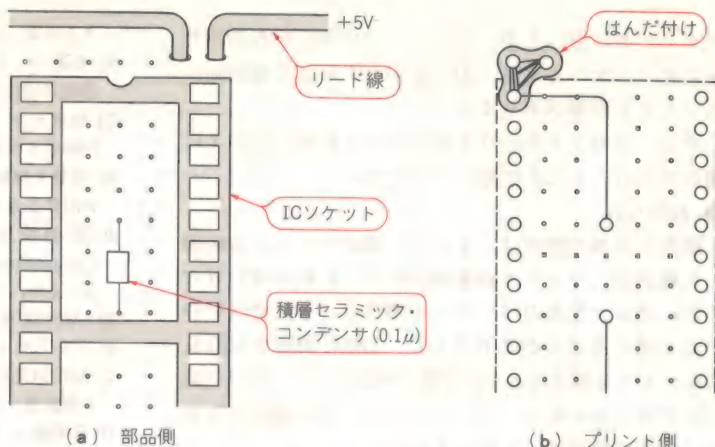
製作について

プリント基板を起こすのがベストですが、今回は時間や予算の関係で、ユニバーサル基板(蛇の目タイプ)を用いて製作しました。大きさは、115mm×150mm程度必要になります。

〈図7〉部品配置例



〈図8〉 パソコンの実装例(RAM, ROMの場合)



部品の配置は、図7を参考にしてください。あまりお勧めできませんが、64K SRAMは、省スペース省電力化のため、3～4個を上下に重ね、 \overline{CE}_1 (20ピン)以外の端子を直接はんだ付けして、24～32KバイトのRAMモジュールのようにして実装しました。

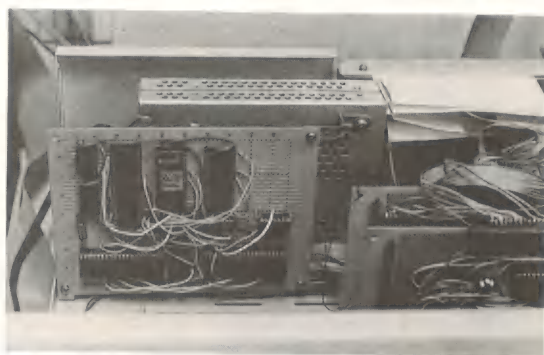
C-MOS SRAMは、発熱が極めて少ないので、信頼性には問題ないようです。なお、RAMに直接はんだ付けをするときは、リーク電流の少ない(セラミック・ヒータ使用の)はんだゴテを用い、素早く行ってください。

配線は、普通のビニル皮覆の撚り線や直径0.3mmのポリウレタン線(ポリウレタンをコーティングした銅線)をはんだ付けして行いました。+5VとGNDライン以外はラッピング配線でもかまいませんが、ICソケットに相当のコストがかかってしまうでしょう。

配線の順序は、ICソケットを固定した後、パソコンの取り付け、+5V、GNDライン、続いて \overline{RD} 、 \overline{WR} などの制御ラインの順に行います。パソコンに積層セラミック・コンデンサを用いれば、図8のようにICソケットの中に収めることができますはずです。また、+5V、GNDラインは、太めの撚り線で短く配線してください。

最後に、アドレス、データ・バスの配線を、基板の裏面でポリウレタン線を用いて行います。ポリウレタン線は、溶けたはんだの熱で皮覆が取れるようになっていますが、その皮覆が溶けるのにやや時間を要するので注意が必要です。そこで、接続する前にポリウレタン線のはんだ付け箇所を、あらかじめはんだメッキして皮覆を溶かしておいたほうが確実です。

配線が一応終わったら、必ずテストなどでソケットの上から導通をチェックしてください。OKならばICを正しく取り付け、+5Vの電圧を加えてください。オシロスコープで、Z80の ϕ に数MHzのクロックが入力されていること、 \overline{INT} 端子が“H”になっていることを確認します。



〈写真2〉 PC8012(拡張I/Oユニット)内に組み込んだ例

データ入力がないのに \overline{INT} が“L”になったままになっている場合は、PPI周辺に配線不良があり、PPIのモード設定がなされていないことが多いようです。

◆ 本機の使用方法

コネクタを入力、出力共に正しく接続し、電源を入れてください。このとき、パソコンやプリンタなどの電源も同時に投入しないと、機種によっては動作しなかったり、誤動作することがあります。

プリンタの切り替えは、例えばBASICでは、

```
LPRINT└chr$(27)+“ア”;
```

または、

```
LPRINT└chr$(27)+“イ”;
```

を実行することにより行われます。“ア”が出力1、“イ”が出力2に対応しており、電源投入時には出力1が選択されています。

ソフトの説明でも述べましたが、ビット・イメージを出力するときは、上記のコントロール・コードと誤解釈される恐れがありますので、あらかじめSW₂を閉じておいてください。その場合、出力ポートの選択はSW₂を閉じる前の状態に固定されてしまうことに注意してください。

リセットSWは、印字を中止し、データをキャンセル

また、リセットを行うと出力ポートが必ず出力 1 に選択されることにも注意してください。

使用してみた感想は、8ビット系のパソコンに接続した場合は、バッファ容量が32Kバイトあれば十分ですが、16ビット系のパソコンの場合は、32Kではやや不足を感じることがありました。Z80を使用する限り、56Kバイトを越えるRAMを扱う場合はバンク切り替えをすることになり、ハード、ソフト共に複雑になるので今回は見送ることにしました。

◆参考·引用·文献◆

- (1)*スター精密, AR-2400取扱説明書。
- (2)*エプソン, VP-130K, IP-130K取扱説明書。
- (3)*日本電気, PC-PR201H2, PC-PR601, PC-PR406LP ユーザーズマニュアル。
- (4)*日本電気, PC-9801 ユーザーズマニュアル。
- (5) 神崎康宏; 特集*マイコン設計技術の完全マスタ, トランジ

(19) 柴田健司；プリンタ，プロッタ切り替え器の製作，トランジスタ技術，1985年4月号，p. 491.

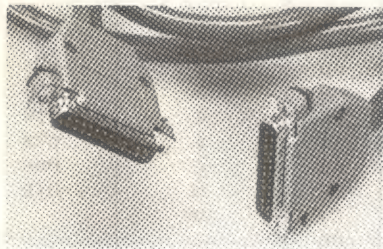
トランジスタ技術
SPECIAL

＜リスト1＞コントロール・プログラム(つづき)

```

183: 09E1 3E 00 LD A,0H ;STROBE ACTIVE
184: 09E3 D3 02 OUT (2H),A ;FC OUT
185: 09E5 3E 0A LD A,0AH ;STB TIMER 1=1M1
CRO SEC.
186: 09E7 3D 00 LD A,NZ,TIMER
187: 09E8 C2 09E7 JP NZ,TIMER
188: 09E9 3E 01 LD A,1H ;STB INACTIVE
189: 09ED D3 02 OUT (2H),A
190: 09EF C3 0940 JP MLOOP
191:
192: ;PRINT OUT CH-2
193: ;
194: ;
195: 09F2 FB 06 IN A,(6H) ;BUSY
196: 09F3 DB 06 AND 40H
197: 09F5 E6 40 JP NZ,POUT2
198: 09F7 C2 09F2 ;
199: ;
200: 09FA 1A LD A,(DE) ;DATA OUT
201: 09FB D3 04 OUT (04H),A
202: ;
203: ;
204: 09FD 3E 00 LD A,0H ;STB ACTIVE
205: 09FF D3 07 OUT (7H),A
206: 0A01 3E 0A LD A,0AH ;STB TIMER
207: 0A03 3D 00 LD A,NZ,TIMER2
208: 0A04 C2 0A03 JP NZ,TIMER2
209: 0A07 3E 01 LD A,1H ;STB INACTIVE
210: 0A08 D3 07 OUT (7H),A
211: 0A0B C3 0940 JP MLOOP
212: ;PRINT OUT CH-3
213: ;
214: ;
215: 0A0E FB 06 IN A,(6H) ;BUSY
216: 0A0F DB 06 AND 80H
217: 0A11 E6 80 JP NZ,POUT3
218: 0A13 C2 0A0E ;
219: ;
220: 0A16 1A LD A,(DE) ;DATA OUT
221: 0A17 D3 05 OUT (05H),A
222: ;
223: ;
224: ;
225: 0A19 3E 02 LD A,2H
226: 0A1B D3 07 OUT (7H),A
227: 0A1D 3E 0A LD A,0AH
228: 0A1F 3D 00 LD A,NZ,TIMER3
229: 0A20 C2 0A1F JP NZ,TIMER3
230: 0A23 3E 03 LD A,3H
231: 0A25 D3 07 OUT (7H),A
232: 0A27 C3 0940 JP MLOOP
233: ;
234: ;
235: ;SUBTTL INTERRUPT ROUTINE
236: ;
237: ;
238: ;
239: ;
240: ;
241: ;
242: ;
243: ;
244: 0A2A 08 INTR: EX AF,AF'
245: 0A2B 03 INC BC

```

§ 2-1

RS-232C インターフェース の基礎

里 和政

最近多くなってきたパソコン通信は、電話回線にモデムを接続することによって行い、パソコンとモデムは、RS-232Cインターフェースで接続します。

またほとんどのパソコンには、RS-232Cインターフェースが標準装備になっていますので、簡単にシリアル伝送が行えます。

● RS-232Cとは

このインターフェースは、米国のEIAがCCITTのV.24, V.28勧告にしたがって定めたシリアル・インターフェースの規格で、信号線の電氣的仕様、信号線の種類およびその機能、機械的特性(コネクタ仕様)などから成っています。そのため、伝送制御手順(通信プロトコル)は、とくに定められていません。

その電氣的特性を表1に、信号線の説明を図1に示します。

規格での最大電源電圧は、 $\pm 15\text{V}$ ですが、実際ではほとんど $\pm 12\text{V}$ を使用しています。従って $\pm 10\text{V}$ ぐらいが信号電圧となります。

● 電圧レベル変換ICが必要

この特性を満足する、専用のドライバ/レシーバICがあります。つまり、TTLレベル($0 \sim 5\text{V}$)をRS-232Cの規定しているレベルへ、電圧を変換するICです。

代表としてドライバには、TI社のSN75188、レシーバにはSN75189Aがあります。図2(a)に構成、図(b)に接続法を示します。

SN75189Aのコントロール端子は、ノイズ除去のコンデンサを付けます。オープン状態で 6V の 10ns のノイズが除去されます。コンデンサは、 12pF から 500pF ものを使用し、 6V のノイズの場合 $15\text{ns} \sim 1000\text{ns}$ 幅で除去できます。

最近では、単一 5V で動作するドライバも出ています。

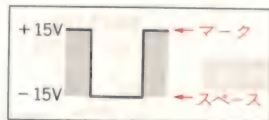
通信手順

伝送制御手順(プロトコル)は、通信するコンピュータ間で同一の手順を使用しなければ、正しく通信することができません。

シリアル・インターフェースでは、1本の線でデータの伝送を行うために、相手がいつデータを送ってきたとか、いつデータが終わったかの伝送のタイミングが不明になります。このデータのタイミングを決める方式に、非同期式と同期式の2種類があります。非同期式は1本線で、同期式は相手の伝送タイミングを取するためにデータ用とクロック用の2本線で行います。

〈表1〉⁽⁴⁾
RS-232Cの電氣的特性

ドライバ出力ロジック・レベル (負荷 $3\text{k} \sim 7\text{k}\Omega$ 時)	$+15\text{V} > \text{oh} > +5\text{V}$ $-5\text{V} > \text{ol} > -15\text{V}$
ドライバ出力電圧(開放時)	$ V_o < 25\text{V}$
ドライバ出力インピーダンス(電源断時)	$R_o > 300\Omega$
出力回路電流(短絡時)	$ I_o < 0.5\text{A}$
ドライバ・スルーレート(立ち上がり特性)	$dv/dt < 30\text{V}/\mu\text{s}$
レシーバ入力インピーダンス	$7\text{k}\Omega > R_{in} > 3\text{k}\Omega$
レシーバ入力電圧	$\pm 15\text{V}$ (ドライバに同じ)
入力開放時のレシーバ出力	マーク("1")
+3V入力時のレシーバ出力	スペース("0")
-3V入力時のレシーバ出力	マーク("1")
ロジック"0" = スペース = 制御ON	$+15\text{V} \sim +5\text{V}$
ノイズ・マージン(雑音余裕度)	$+5\text{V} \sim +3\text{V}$
過渡領域(入力電圧の無効域)	$+3\text{V} \sim -3\text{V}$
ノイズ・マージン	$-3\text{V} \sim -5\text{V}$
ロジック"1" = マーク = 制御OFF	$-5\text{V} \sim -15\text{V}$

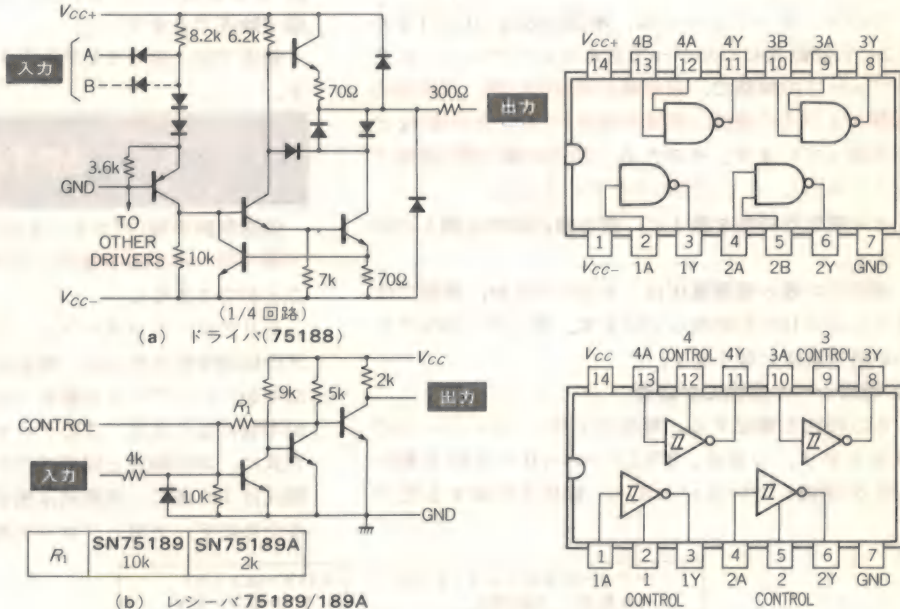


〈図1〉⁽⁴⁾
信号線の説明

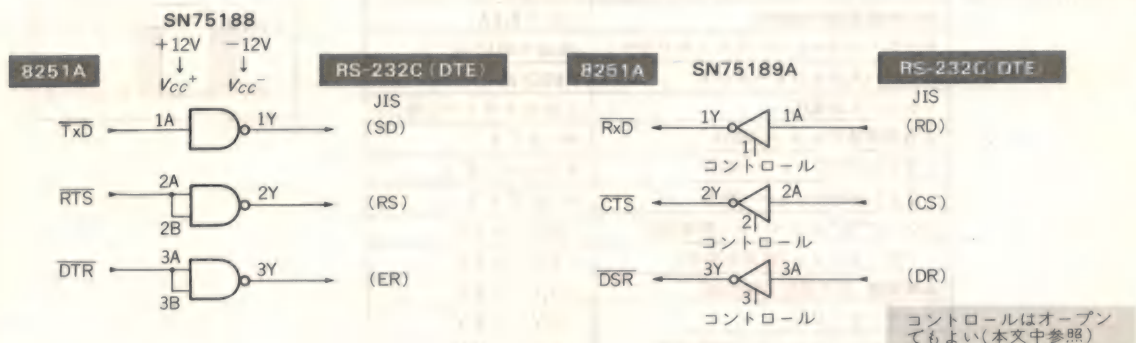
名 称	略 号			ピン番号 (25ピン・コネクタ)	8251Aに接続さ れるピンの名称
	RS-232C	CCITT	JIS		
保安用接地	AA	101		1	
信号用接地	AB	102	SG	7	
送信データ	BA	103	SD	2	TxD
受信データ	BB	104	RD	3	RxD
送信要求	CA	105	RS	4	RTS
送信可	CB	106	CS	5	CTS
データ・セット・レディ	CC	107	DR	6	DSR
データ端末レディ	CD	108/2	ER	20	DTR
被呼表示	CE	125	CI	22	
データ・チャネル受信キャリア検出	CF	109	CD	8	
データ信号品質検出	CG	110	SQD	21	
データ信号速度選択	CH/CI	111	SRS	23	
送信信号エレメント・タイミング*	DA/DS	113/114	ST ₁ /ST ₂	24/15	
受信信号エレメント・タイミング	DD	115	RT	17	
従局送信データ	SBA	118	BSD	14	
従局受信データ	SBB	119	BRD	16	
従局送信要求	SCA	120	BRS	19	
従局送信可	SCB	121	BCS	13	
従局受信キャリア検出	SCF	122	BCD	12	

(*) 送信信号エレメント・タイミングには、データ端末装置→モデム(DA)と、モデム→データ端末(DS)の2種類がある

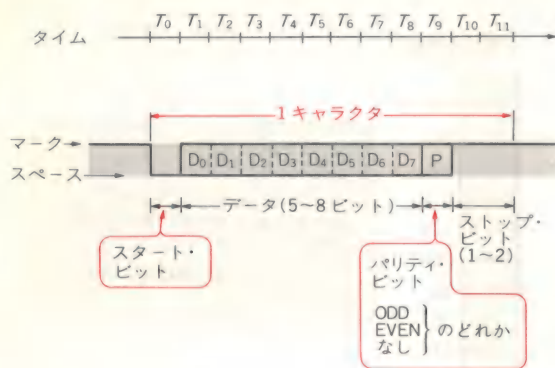
〈図2(a)〉⁽¹⁵⁾
代表的RS-232Cドライバ
/レシーバ



〈図2(b)〉 SN75188とSN75189Aの接続図



〈図3〉非同期式伝送のフォーマット



非同期式 (ASYNC : Asynchronous Communication)

図3に非同期式の伝送フォーマットを示します。

シリアル伝送の場合、基本となる同期クロックが必要になりますが、このクロックをもとにして伝送するデータのタイミングを決定します。

この伝送のタイミング(速度)を、ボーレートと呼びます。ボーレートは、1秒間に何ビットのデータを送れるかの数を示します。300ボーでは、1秒間に300ビット、1200ボーでは、1200ビットとなります。

この方式では、1キャラクタごとに同期を取るために、スタート/ストップ・ビットがキャラクタごとに付加されます。

スタート・ビットは1ビットですが、ストップ・ビットは1, 1.5, 2ビットの場合があります。図4に非同期式の仕様を示します。

したがって、非同期式の場合には常に2~4ビットの余分なビットがキャラクタごとに付加されるために、伝送効率が悪くなります。

しかし、この方式では、無手順と呼ぶたれ流しの方法を用いると、伝送の手順を簡単にすることができず。

また、非同期式のことを調歩同期式とも呼びます。

図5(a)~(c)にターミナル(DTE)とターミナル(DTE)の接続を示します。ときには、この接続をT-T結線と呼びます。

図5(d)は、モデム(DCE)とターミナル(DTE)の接続を示します。この接続は、M-T結線と呼びます。

M-T結線の場合は、ストレートの接続になりますが、モデムからの信号線の出し方に特徴があり、SD信号は、ターミナルでは出力となっていますが、モデムでは入力になり、伝送コントローラのRxD(受信)に接続されます。以下の信号も同様に入出力がターミナルの逆になっています。

とくにこの部分で接続の問題が発生しますから、接

〈図4〉代表的な非同期式の仕様例

機能	仕様
ボーレート	75, 150, 300, 600, 1200, 2400, 4800, 9600, 19200
文字長	5, 6, 7, 8ビットのいずれか
パリティ	ODD, EVEN, なしのいずれか
ストップ・ビット	1, 1.5, 2ビットのいずれか
通信方式	全二重 または 半二重

続する相手のRS-232Cの型式がDTEかDCEかを調べる必要があります。

同期式 (SYNC: Synchronous Communication)

図6に同期式の伝送フォーマットを示します。

同期式では、非同期式では必要だったスタート/ストップ・ビットがなくなるため、伝送効率が上がります。

しかし、同期を取るために同期キャラクタ (SYN) がデータとは別に必要になります。同期キャラクタは、1キャラクタごとに付加するのではなく、データ・ブロックごとに付加します。一度同期を取ることによって、そのブロックが終了するまで同期はとれています。このブロックの長さは、256バイトまでにしている場合が多いようです。それ以上のときは、複数ブロックに分けて送信します。

パイシニングの手順

同期式の通信プロトコルとして、パイシニング (BSC) がよく使用されますが、同期コードが2文字以上必要です。

受信側では、同期キャラクタを受け取って、同期を確立してから実データの受信を行います。

一度確立した同期は、ブロックの終わりまで保持されますが、途中でデータが途切れた場合、同期は保証されません。そのために、途中でデータが切れるときには、同期を保持するためにSYNコードを送ります。このSYNコードをタイム・フィラ (Time Filler) と呼びます。受信側では、このSYNコードを除去します。

この方式では、伝送を制御コードによって制御する必要があります。図7に同期式のブロック・フォーマットを、図8に制御コードの一覧を示します。

しかし、制御コードを使用しているために、すべての文字を送ることができません。そのために、このような文字も送ることのできる透過モードがあります。

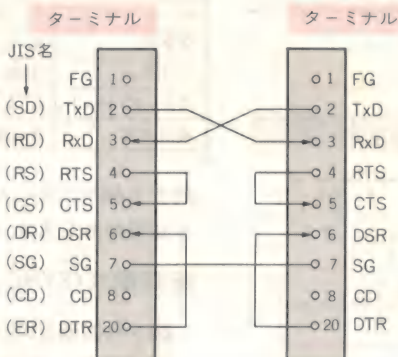
透過モードは、伝送制御コードの前にDLEコードを付加して、制御コードとは区別します。DLEコードをデータとして送るときは、DLE, DLEとします。

図9にコネクタの接続方法を示します。図(a)は、ターミナル(DTE)同士の接続です。同期式の場合、モデムとターミナル間で同期を確実にとるために、送信クロック線を相手の受信クロック線に接続します。

図(b)は、モデム(DCE)とターミナル(DTE)を接続する場合です。

〈図5〉 ターミナルとモデム/ターミナルの接続例

(a) ターミナル-ターミナル(i)

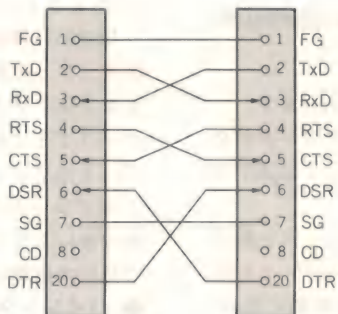


送信-受信
受信-送信)を接続

それぞれの端末は、アクティブになったら相手の状態に無関係に送受信可能となる。

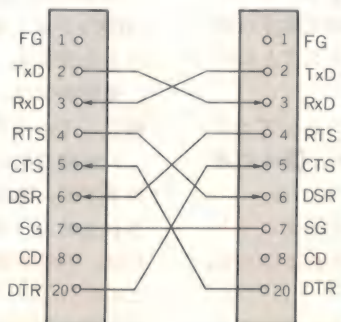
データの送受信の確認をソフトウェアで行う

(b) ターミナル-ターミナル(ii)



お互いのRTS-CTSを接続することで、送信要求に対して送信可を得ている。
端末タイプ同士の接続は多くの場合この形になっている。
しかし、厳密な意味でのハンドシェイクは困難である。
交互の通信のときに特に必要なのは、相手側が受信可であるかどうかを知ることであるが、この接続では、送信要求で相手側を送信可にしている。このことを注意してプログラムを作成する必要がある

(c) ターミナル-ターミナル(iii)



DSRをチェックすることで、相手側が送信要求を出しているかどうかを確認できる

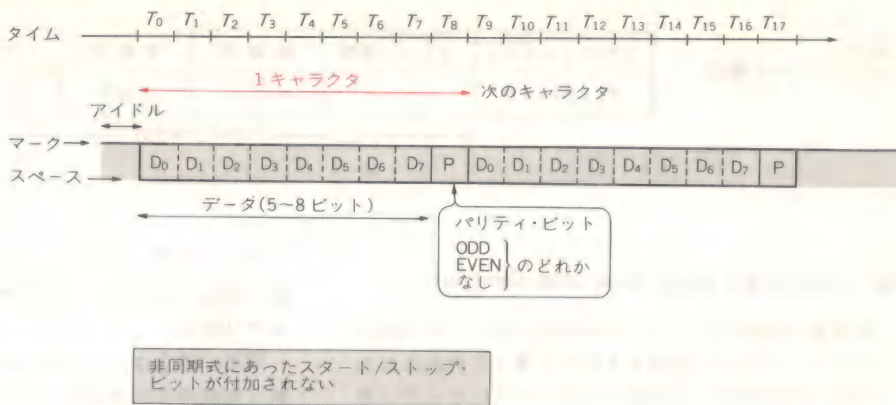
DTRをONにすることで、相手側に受信可であることを知らせ、相手側が送信することができる

(d) モデム-ターミナルの接続

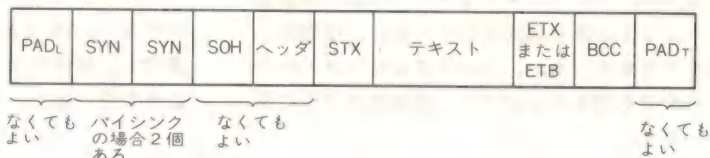


ターミナルでCD,CIが接続されていない場合がある

〈図6〉
同期式の伝送
フォーマット



〈図7〉
同期式のブロック・
フォーマット

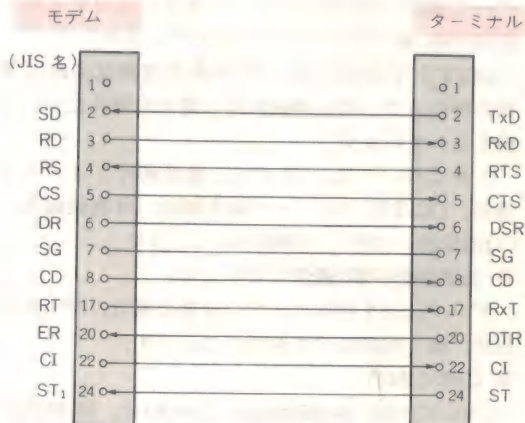
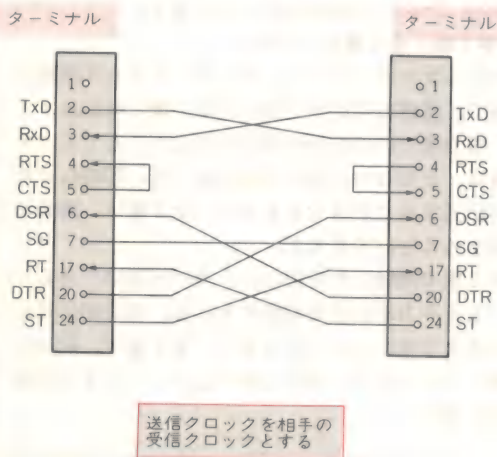


〈図8〉
同期式制御
コードの一覧

制御コード	説 明	ASCII	EBCDIC	説 明
SYN	同期の確立	16H	32H	同期を確立して伝送を開始する
DLE	制御コードの拡張	10H	10H	透過モードなどの拡張制御コード
SOH	ヘッダの始め	01H	01H	局番などのヘッダ部の開始を示す
STX	テキストの始め	02H	02H	データ・テキストの開始を示す
ETB	ブロックの終わり	17H	26H	ブロックの終わり(テキストの集まり)
ETX	テキストの終わり	03H	03H	データ・テキストの終わり
EOT	伝送の終わり	04H	37H	伝送が終了したことを示す
ENQ	接続要求	05H	2DH	ターミナルの接続要求
NAK	否定応答	15H	3DH	伝送に対しての否定応答
ACK ₀	偶数ブロックの肯定応答	10H・30H	10H・70H	伝送に対しての肯定応答
ACK ₁	奇数ブロックの肯定応答	10H・31H	10H・61H	伝送に対しての肯定応答

〈図9(a)〉 同期式ターミナル-ターミナルの接続

〈図9(b)〉 同期式モデム-ターミナルの接続



〈図10〉
HDLCのフレーム構成

フラグ・シーケンス	アドレス情報	制御部	情報部	FCS	フラグ・シーケンス
01111110	8ビット	8ビット	可変長	16ビット	01111110

← FCSの対象範囲 →

HDLC(High level Data Link Control)

非同期、同期には、いくつかの点において欠点があるために、これらを克服するために考えられたのがこのHDLC手順です。IBM社のSDLCが元になっています。

HDLC方式は、フレーム・フォーマットから構成され、フレームごとに誤り制御を行うために、信頼性の高い伝送ができます。また、どのようなビット・パターンのデータでも送ることができ、透過性は完全に保証されています。

フレームは、フラグ・シーケンス、アドレス情報、制御、情報、FCSおよびフラグ・シーケンスから成っています(図10)。

同期の確立は、フラグ・シーケンスによって行われ、誤り制御用のFCS(フレーム・チェック・シーケンス)フィールドは、16ビットのCRCで行われます。

これによって、HDLCは、高速で信頼性の高い伝送ができ、LANなどのコンピュータ・ネットワークに最適です。具体的なプログラム例はZ80 SIOの解説のところで行います。

用語解説

● CCITT V.24, V.28

CCITT V.24勧告は、「データ端末装置(DTE)とデータ回線終端装置(DCE)間の相互接続回路の定義」という課題で、同期、非同期通信、専用回線でのデータ通信、交換網でのデータ通信に使用する相互接続回路、動作条件について定めています。

CCITT V.28勧告は、「不平衡復流相互回路の電気的特性」で、相互接続回路の電気的特性について規定されています。

これについては、「モデムと電話網によるデータ通信、CCITT Vシリーズ勧告解説、林高雄編著、CQ出版社」に詳しく解説されています。

● 伝送制御手順(通信プロトコル)

データ伝送を行ううえでの同期の確立、データの形式、誤り制御などを決めることです。

● CRCとBCC

CRC(Cyclic Redundancy Check)は、同期式伝

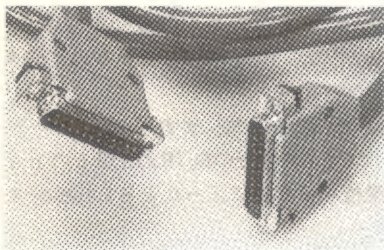
送でよく利用される誤り検出方法のひとつです。

この方式は、送信するデータを数値(メッセージ多項式と呼ぶ)として、あらかじめ決められた値(生成多項式と呼ぶ)で割ります。その余りがBCC(Block Check Character)となります。

● 半2重と全2重について

モデムなどのシリアル・インターフェース装置の仕様で、通信方式が半2重とか全2重といわれますが、双方向で伝送を行う場合の方式のことです。半2重は、どちらか一方のみが伝送でき、同時にお互いが伝送することはできません。全2重は、同時に伝送することができます。

半2重で問題となるのは、一方が伝送中であるとき、もう一方が伝送を中断するには、相手が伝送を終わるまで待つことになります。全2重で、伝送を中断したいときは、相手に割り込むことにより可能となります。



§ 2-2

シリアル・インターフェース用 LSIの使い方

里 和政/相良富美/斉藤健司

シリアル・インターフェース用のLSIには、8251A、Z80 SIO、6850などがあります。表 1 に80系LSIの比較を示します。

8251A (USART)

8080/8086系のデータ・バス 8 ビット用シリアル・インターフェースLSIで、最も利用の多いICです。

8251Aの端子の機能

8251Aは、図 1 で示されるような28ピン構成で、図 2 のブロック・ダイアグラムで示される以下のような機能をもっています。

- (1) データ・バスとリード/ライト制御
- (2) モデム制御

(3) 送信バッファと送信制御

(4) 受信バッファと受信制御

以上のような構成となっています。

端子機能を次に示します。

► $D_0 \sim D_7$ システム側のデータ・バス信号線と接続する端子です。

► C/\overline{D} 8251Aのデータ・バス上の情報がデータかコントロール・ワードまたはステータス情報であるかを区別し、“H”のときコントロール、“L”のときデータを示します。

► \overline{RD} この入力信号線が“L”のとき、8251Aからデータやステータス情報を読み出すことを示します。

► \overline{WR} CPUから8251Aにコントロール・ワードや、データを書き込む信号で、“L”入力アクティブです。

► \overline{CS} この入力信号が“L”のとき、8251Aをイネー

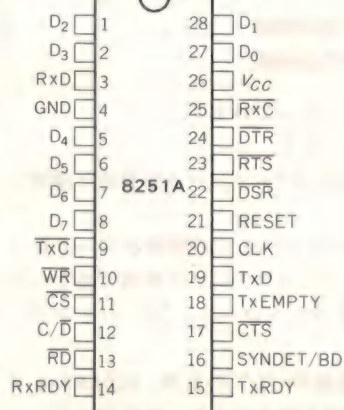
〈表 1〉
各通信LSIの比較(80系)

種類 比較項目	8251A	Z80 SIO	μ PD7201
ピン数	28	40	40
チャンネル数	1	2	2
データ・レート	~9600bps(非同期) ~56kpbs(同期)	~55kbps(非同期) ~880kbps(同期)	~55kbps(非同期) ~880kpbs(同期)
非同期モード キャラクタ長 クロック ストップ・ビット パリティ・ビット 受信バッファ数	5~8 $\times 1, \times 16, \times 64$ 1, 1.5, 2 Even, Odd, なし 1	5~8 $\times 1, \times 16, \times 32, \times 64$ 1, 1.5, 2 Even, Odd, なし 3	5~8 $\times 1, \times 16, \times 32, \times 64$ 1, 1.5, 2 Even, Odd, なし 4
同期モード 同期 CRC 同期キャラクタ	内部, 外部 なし 1 or 2	内部, 外部 あり 1 or 2	内部, 外部 あり 1 or 2
HDLモード CRC NRZI		あり なし	あり なし
割り込みモード	$TxRDY$ $RxRDY$ (Z80 モード 1)	Z80 モード 2	8085A モード 8086モード Z80 モード 1
DMA機能	なし	なし	あり(制限付き)
内部レジスタ	2 (パラメータ) 1 (ステータス)	8 (パラメータ) 3 (ステータス)	8 (パラメータ) 3 (ステータス)
メーカー名	インテル, 日本電気, 沖電気, 三菱, 東芝など	サイログ, シャープ 東芝, ロームなど	日本電気など

▶ $\overline{\text{TxC}}, \overline{\text{RxC}}$ 送受信キャラクタのボーレートを制御するクロックで、同期式の場合には実際のボーレート

▶ RxRDY コマンド命令のD₂(RxE)が1のときでキャラクタが受信バッファに入ったときにReadyを表示

ピン接続図



種類	ピン番号	ピン名	信 号 名	I/O	種類	ピン番号	ピン名	信 号 名	I/O
データ・バス	1, 2 5~8 27, 28	D0~D7	データ・バス	I/O	トランスミッタ	19	TxD	Transmit Data	O
	21	RESET	リセット	I		15	TxRDY	Transmitter Ready	O
	20	CLK	クロック	I		18	TxE	Transmitter Empty	O
	10	\overline{WR}	Write Data or Control	I	レシーバ	9	$\overline{Tx\overline{C}}$	Transmitter Clock	I
	13	\overline{RD}	Read Data	I		3	RxD	Receive Data	I
	12	C/\overline{D}	Control/Data	I		14	RxRDY	Receiver Ready	O
	11	\overline{CS}	Chip Enable	I		25	$\overline{Rx\overline{C}}$	Receiver Clock	I
22	\overline{DSR}	Data Set Ready	I	16		SYNDET /BRKDET	Sync Detect /Break Detect	I/O	
24	\overline{DTR}	Data Terminal Ready	O	26		V_{cc}	+5V	—	
モデム・コントロール	17	\overline{CTS}	Clear to Send Data	I		4	GND	グラウンド	—
	23	\overline{RTS}	Request to Send Data	O					

8251Aの内部ブロック図



し、割り込み信号などに使用します。

▶ **TxDY** データの転送がReadyになったことを表し、この信号線は、データ・バッファが空で、TxENが1でかつCTSが0のときアクティブとなります。

▶ **DSR** モデムの状態をテストするData Set Readyとして使用します。

▶ **DTR** モデムのData Terminal ReadyまたはRate Selectとして使用します。

▶ **RTS** モデムのRequest to Send dataとして使用します。

▶ **CTS** モデムのClear To Sendとして使用します。

▶ **TxE** “H”のとき、送信バッファから送信キャラクタがなくなったことを示します。

▶ **SYNDET/BD** この端子は、同期式モードのときに使用します。この信号線が、モード設定のときに出力として使用されると、受信モードの場合には、あらかじめ定められたSYNCキャラクタを受信すると“H”となります。

入力として使用すると、8251Aは、データ・キャラクタを、次のRxCの立ち下がりでアセンブルを開始します。

また、出力に使用した場合には、モード設定によって定められたビット数のデータが、すべてゼロの状態を検出したとき“H”となります。

▶ **V_{CC}** +5Vを接続します。

▶ **GND** グラウンドを接続します。

● 送受信を行うとき気をつけなければならない動作

CPUによって8251Aにセットされたパラレル・データは、モード命令で指定したシリアル・データに変換され、TxDの出力端子から送り出されます。

ただし、送信データが送り出される条件は、コマンド命令で送信イネーブルとなっており、CTSが“L”

のときです。また、最初の送信データが送り出されるまで、この端子はマーク状態になっています。

受信バッファは、RxD端子からシリアル・データを受信し、パラレル・データに変換してから、ステータスのRxRDYビットをセットします。

このときフレーミングとパリティのチェック(パリティがイネーブルのとき)を行い、エラーがあるとステータスにセットします。受信データをCPUに入力するときは、ステータスを入力してRxRDYがセットされていることを確認して実行します。

受信データをCPUに入力すると、RxRDYはリセットされますが、エラーが発生した場合には、コマンド命令を実行してリセットします。

■ 8251Aのプログラミング

8251Aの設定は、まず図3で示すモード・レジスタによって動作モードを決めます。非同期モードには、伝送速度ファクタ、伝送キャラクタ長、パリティの有無およびストップ・ビット数(1, 1.5, 2ビットのいずれか)があります。

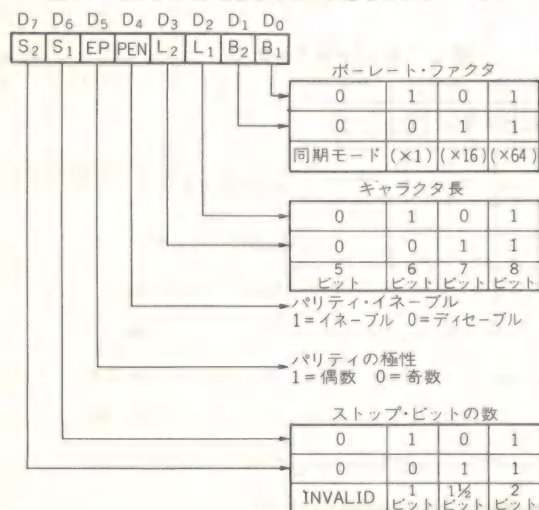
● 同期モードの設定

図4は、同期モードでのモード・レジスタです。SCSはSYNコードの数、ESDはSYNコードの検出を外部で行うか、8251Aで行うかを選択します。

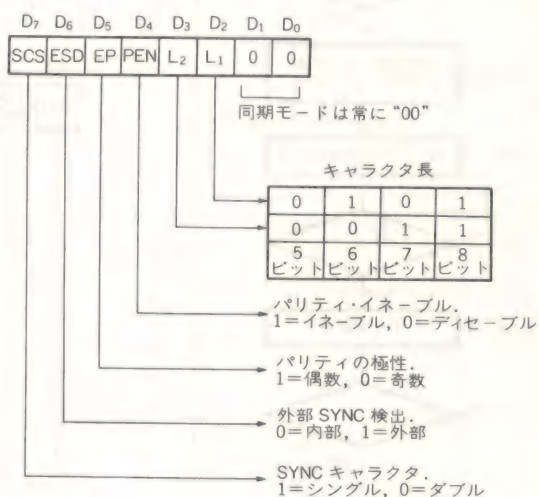
伝送速度は、TxC, RxCに供給したクロックを分周して決められます。その分周比が伝送速度ファクタです。19.2kHzをクロックとした場合、1×で19200ボー、16×で1200ボー、64×で300ボーとなります。

通常は、16×がよく使用されています。ただしこれは、非同期モードで変えることができ、同期モードのときは1×の固定です。図5は、8251Aの動作フローチャートです。

〈図3〉⁽⁷⁾ 8251Aの非同期でのモード命令のフォーマット



〈図4〉⁽⁷⁾ 8251Aの同期式でのモード命令フォーマット



● コマンド・レジスタの設定

図6にコマンド・レジスタを示します。コマンドによって伝送制御を行います。TxEは送信を行うときに“1”，RxEは受信を行うときに“1”にします。SBRKは，TxDを強制的に“L”にします。

ERは，受信エラー（PE，OE，FE）をリセットします。IRは，内部リセットしてモードを再設定にします。

EHは，ハント・モードに入るときに“1”にします。ハント・モードとは，同期モードにおいてSYNコードのみを受信することをいいます。

図7にステータス・レジスタを示します。ステータス・レジスタは，送信/受信の状態を確認します。TxRDYとTxEによって送信バッファが空になると“1”になり，TxEは送信シフト・バッファが空になると“1”になります。

PE，OE，FEは，受信エラーの状態を表します。PEは，パリティ・エラーがあったときにセットされます。OEはオーバラン・エラーと呼び，受信バッファにデータがあり，続けて受信して前のデータが破壊されたときにセットされます。FEはフレーミング・エラーと呼

び，ストップ・ビットが正常に認識されないときセットされます。

SYNDETは，同期モードではSYNコードを検出したときにセットされ，非同期モードでは，ブレーク・キャラクタを検出したときにセットされます。

以上のレジスタにより伝送します（リスト1参照）。

注意点としては，RESET動作で6クロックかかり，モード・コマンドでは，非同期の場合8クロック，同期の場合16クロック分ICの内部処理がかかります（1クロック 最大3.125MHz）。

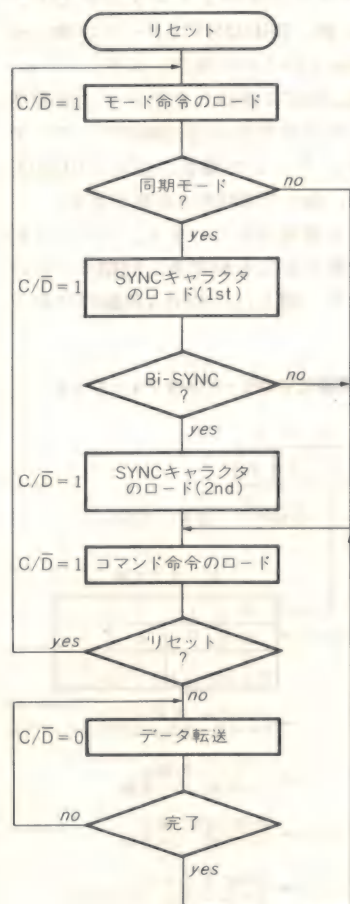
そのため，高速のCPUを使うときには，ウェイトが必要になります。

8251Aは，外部から送受信クロックを入力するために，そのクロックを8253（プログラマブル・タイマ）などを使用すると容易に，ボーレートを変更することができます。

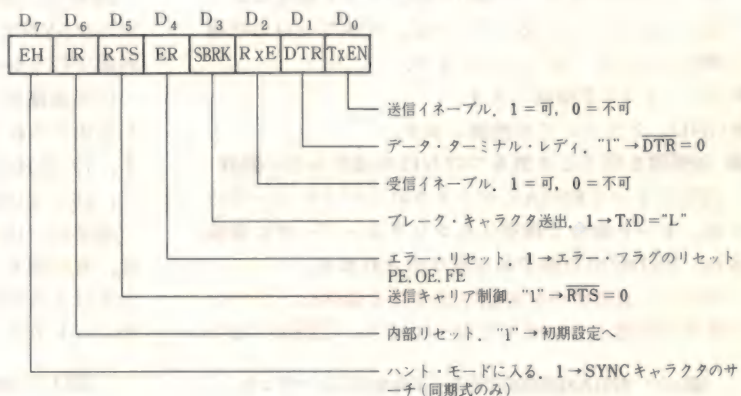
6850/6350 (ACIA)

ACIAは，その名のとおり調歩同期式直列インター

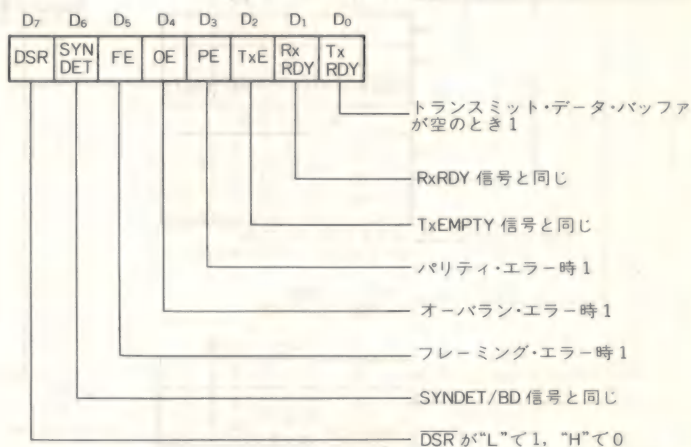
〈図5〉⁽¹⁾⁽⁷⁾ 8251Aの初期設定フローチャート



〈図6〉⁽⁷⁾ コマンド・レジスタ



〈図7〉⁽⁷⁾ 8151Aのステータス



<リスト1>
 8251A による
 シリアル入出
 力
 (Lattice C)

```

/*
  U S A R T - 8 2 5 1 A   シリアル伝送 (非同期)   プログラム

*/

#define csreg 0x00      /* 8 2 5 1 A   コントロール / ステータス */
#define dtreg 0x01      /* 8 2 5 1 A   データ   ポート */

#define reset 0x40      /* リセット コマンド */
#define mode 0x04e      /* モード設定 コマンド */
                        /* クロック分周比 1 / 16 */
                        /* データ・ビット長 8ビット */
                        /* ストップ・ビット 1ビット */
                        /* パリティなし */
#define rvcmd 0x6        /* 受信コマンド */
                        /* 受信イネーブル */
                        /* D T R   オン */
#define sdcmd 0x23      /* 送信コマンド */
                        /* 送信イネーブル */
                        /* D T R   オン */
                        /* R T S   オン */
                        /* エラーリセット */
#define eroff 0x10      /* レシーブ・レディまたはエラー */
#define rvsts 0x03a      /* レシーブ・エラー */
#define ersts 0x038      /* センド・レディ */
#define sdsts 0x01      /* 送受信可能 */
#define dsr 0x080       /* 初期化 ルーチン */

sinit()
{
    int i;

    for(i=0; i=4; i++)      /* ダミー コマンド出力 */
    {
        outp(csreg, 0x83);
    }
    outp(csreg, reset);      /* コントローラ リセット */
    outp(csreg, mode);      /* コントローラ モード設定 */
    return(0);
}

/* データ 受信 1 バイト */

data_in(data)
char data;
{
    int stat;

    outp(csreg, rvcmd);      /* 受信モード設定 */
    /* データ受信まで待つ */
    while((stat=inp(csreg)&rvsts)==0) {}

    if((stat&ersts)!=0)
    {
        outp(csreg, rvcmd+eroff); /* エラー・ビット リセット */
        return(-1);             /* エラー 終了 */
    }

    data = inp(dtreg);        /* データ受信 */
    return(0);
}

/* データ バイト 送信 */

data_out(data)
char data;
{
    int stat;

    outp(csreg, sdcmd);      /* 送信モード設定 */
    while(((stat=inp(csreg))&sdsts)==0) /* ステータス リード */
    {
        if((stat&dsr)==0)
        {
            /* 送信不可 */
            return(-1);
        }
    }

    outp(dtreg, data);        /* データ 送信 */
    return(0);
}

```

フェースですから、8251Aのように同期式との切り替えはありません。

もともと Bell103, 600bps モデムである MC6860 と組んで、電話回線で長距離データ通信ができるように設計されています。

図8にピン配置を示します。

● MPUがアクセスできるACIAのレジスタ

ACIAの内部には、MPUがアクセスできる4本のレジスタがあります。2本ずつ同じ内部アドレスに配置されています。図9にACIAの内部ブロック図を示します。

受信データ・レジスタとステータス・レジスタは読み出し専用ですし、送信データ・レジスタとコントロール・レジスタは書き込み専用になっていて、ACIAが一度に必要とするアドレスは2バイトだけです。

● コントロール・レジスタで付加ビットを設定する

直列伝送を行うためには並列で扱っているデータを直列に、またその逆の変換をしなければなりません。簡単に思い付くのはシフトレジスタですが、実際にACIAにはシフトレジスタが入っているのです。ただたんに、並列-直列変換をしてもどこが始まりで終わりのか受信側はわかりません。ですから送信側では、直列データにスタート、ストップ・ビットを付加してやりま

す。データの伝送は長距離を想定しているの

で、雑音が入らないとも限りません。そこで受信データの誤りを識別できるようにパリティ・ビットも付けられるよう

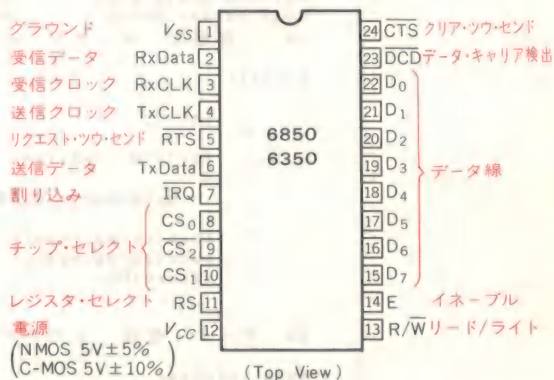
にします。パリティ・チェックですから、CRCのように正確ではありません。これらの付加ビットは、コントロール・レジスタの設定によって行います。

● 転送速度の決め方

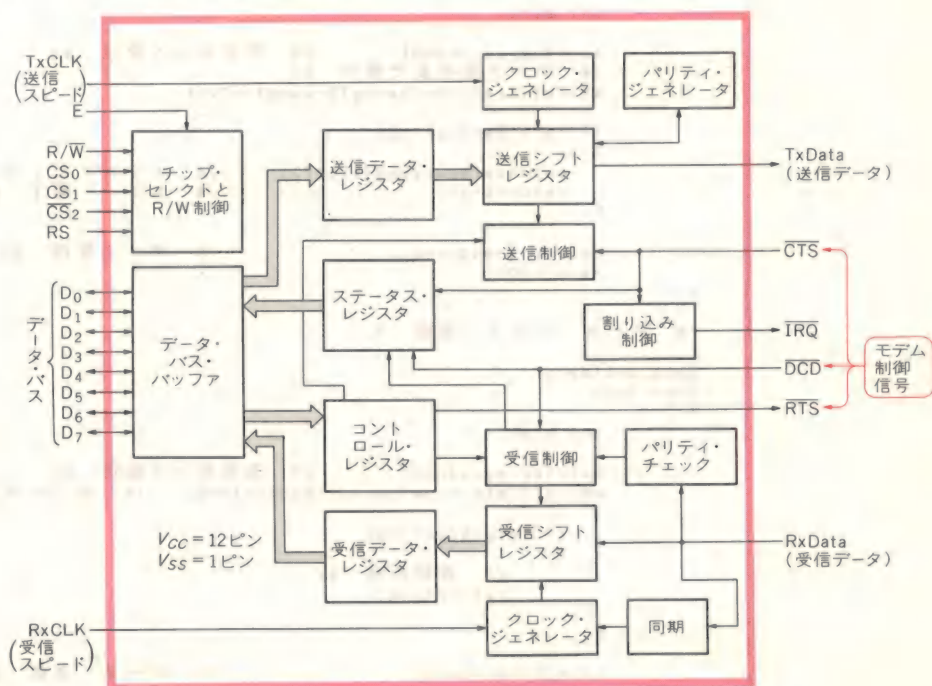
並列→直列変換には、一定のクロック・パルスが必要です。周知のようにデータ通信では、転送速度を決めるボーレートというのがあり、これを設定するためには、シフトレジスタのクロック・パルスに相当する RxCLK と TxCLK が必要です。

ACIAでもソフトウェアでボーレートを変化させられるように、コントロール・レジスタの設定によって入力クロックを3段階に分局できます。なお、ACIAが扱える最大のボーレートは、Bバージョンで1000 kbps までです。

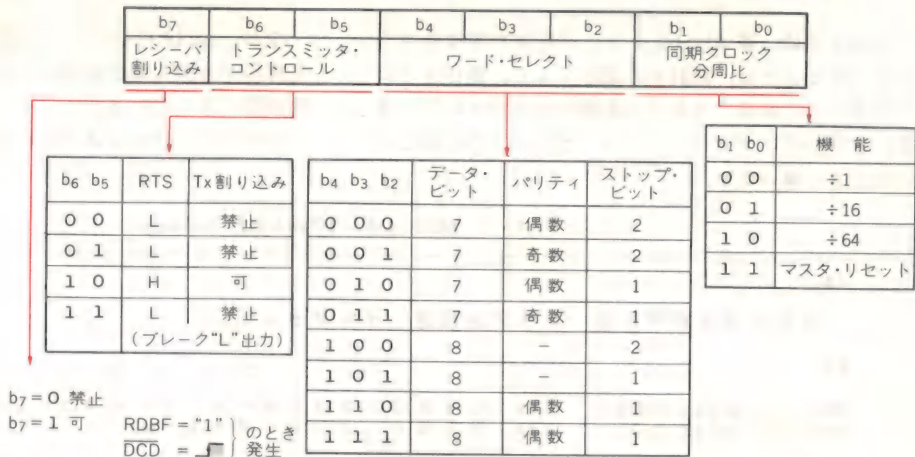
〈図8〉⁽¹⁰⁾ 6850 (ACIA) のピン配置



〈図9〉⁽¹⁰⁾ 6850 (ACIA) の内部ブロック図



〈図10〉⁽¹⁰⁾
6850 (ACIA) のコントロール・レジスタ



● ステータス・レジスタはシフトレジスタの状態を示す

さて、直列伝送ではビット・データを1個ずつ送りますし、その転送速度によって、システム・バス内でやりとりされるスピードに比べてかなり遅くなっています。したがって、前に送ったデータがACIA内のシフトレジスタに残っていれば、次のデータは書き込めません。

受信データも完全にシフトレジスタに入ってしまったら読めず、読み出すことはできません。このために、ACIAでもステータス・レジスタでこの情報を提供しています。

■ ACIAのデータ送信/受信のシーケンス

図10にACIAのコントロール・レジスタの内容を示します。8251Aのようにデータ・ビット長にバリエーションはありませんし、ストップ・ビットの設定に1.5という半端なものもありませんので、RTTYなどの特殊な用途には使えないことになります。また、図11にステータス・レジスタの内容を示します。

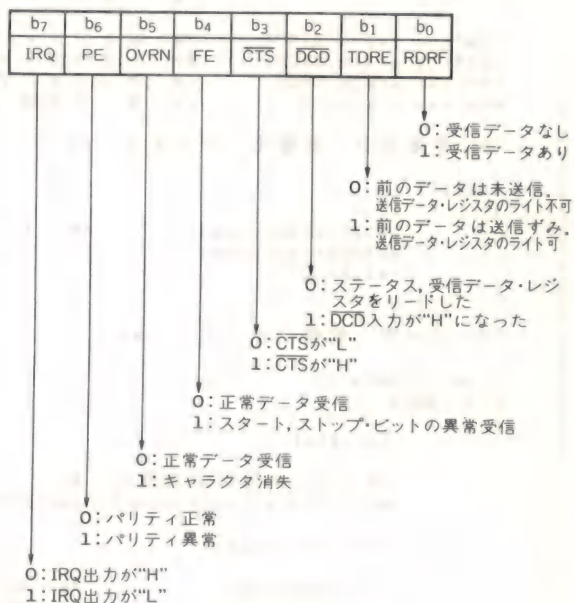
その他に、通信用のLSIの機能として必要なものとして、モデム制御があります。つまり、回線からデータが入力されたときはいつでもMPUはこれに対応しなければなりません。そこでモデムは回線のキャリアを検出しACIAに知らせます。

ACIAはこれを受けてステータス・レジスタのDCDフラグをセットし、もしIRQイネーブルであればMPUに対し割り込みをかけます。また、回線にデータを出力する場合には、モデムにこの要求を出します。回線が使用可能ならモデムは許可信号をACIAに出します。これでACIAは直列データを出力します。

■ ACIAの割り込み

さて、ACIAの割り込みについて見てみます。直列データ転送は、一般に低速でありMPUがこのスピー

〈図11〉⁽¹⁰⁾ 6850 (ACIA) のステータス・レジスタ



ドで処理したのでは、システム全体の処理能力がたいへん低くなってしまいます。

また、いつ入ってくるのか予測のつかないデータをいつまでも待つわけにもいきません。ACIAの割り込み要因を列举してみます。

- ▶ 送信レジスタが空で、モデムからの送信許可があるとき
- ▶ モデムが受信キャリアを検出したとき
- ▶ 受信レジスタにデータがきちんと入ったとき
- ▶ 受信データがオーバーランしたとき

これらは、もちろんコントロール・レジスタの割り込みイネーブル・ビットがセットされていなければ、ハード的な割り込みは発生せず、ステータス・レジスタに結果が残るだけになります。

このように、周辺LSIはシステムのむだをなるべく少なくすることと、設計者の意志により、動作をプログラミングで変更できるよう配慮されて作られています。ですから、ディスクリートで作ったI/Oとはこの辺が大きく違います。

なお、ACIAにはリセット端子がありません。初期化操作が必要な場合には、コントロール・レジスタの再設定により行います。

プログラム例をリスト2に示します。

〈リスト2〉 6850によるシリアル入出力(Lattice C)

```

/*
ACIA-6850 シリアル伝送 プログラム
*/

#define aciactl 0x00 /* 6850 コントロール/ステータス */
#define aciadr 0x01 /* 6850 データ ポート */

#define reset 0x03 /* 6850 リセット コマンド */
#define mode 0x015 /* 6850 モード設定 コマンド */
/* クロック分周比 1/16 */
/* データ・ビット長 8ビット */
/* ストップ・ビット 1ビット */

#define rvsts 0x071 /* レシーブ・レディまたはエラー */
#define ersts 0x070 /* レシーブ・エラー */
#define sdsts 0x01 /* センド・レディ */
#define cts 0x08 /* センド可能 */

/* 6850 初期化 ルーチン */

sinit()
{
    outp(aciactl,reset); /* コントローラ リセット */
    outp(aciadr,mode); /* コントローラ モード設定 */
    return(0);
}

/* データ 受信 1 バイト */

data_in(data)
char data;
{
    int stat;

    /* データ受信まで待つ */
    while((stat=inp(aciactl)&rvsts)==0) {}

    if(stat==ersts)
    {
        inp(aciadr); /* エラー・ビット リセット */
        return(-1); /* エラー 終了 */
    }
    data = inp(aciadr); /* データ受信 */
    return(0);
}

/* データ バイト 送信 */

data_out(data)
char data;
{
    int stat;

    while(((stat=inp(aciactl))&cts+sdsts)==0) {} /* ステータス リード */

    if(stat==cts) /* 送信可能か? */
        return(-1); /* 送信不可 */
    outp(aciadr,data); /* データ送信 */
    return(0);
}

```


Z80 SIO

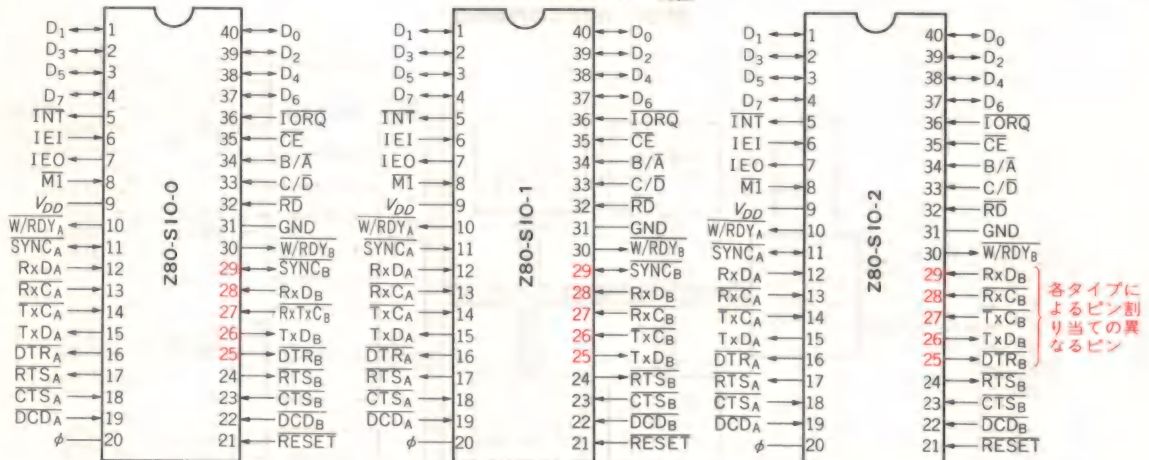
Z80 SIOは、Z80用のLSIとして設計され、非同期、同期、SDLC(HDLC)通信を制御できるLSIです。伝送用のチャンネルは二つです。

● Z80 SIOの特徴

8251Aでは、同期式の場合CRCの計算などを、ソフトウェアによって行う必要がありますが、このLSIはこれらの機能が内蔵されています。

伝送速度は、Z80Aを4MHzクロックとしたとき800kビット/秒です。

〈図12〉^(a) Z80 SIOのピン配置



〈表2〉^(a) Z80 SIO信号の説明

端子名	I/O	機能
D ₀ ~D ₇	双方向	Z80 CPUのデータ・バスと接続し、CPUとSIOの間のデータ転送に使用する
B/ \bar{A}	入力	SIOのAまたはBチャンネルを選択。通常、CPUのアドレス線A ₁ を接続する
C/ \bar{D}	入力	この信号は、CPUとSIO間で転送される信号が、制御コマンドであるかデータであるかを区別する。通常、CPUのアドレス線A ₀ を接続する
\bar{CE}	入力	チップ・イネーブル信号でアドレスをデコードした信号を接続する
ϕ	入力	内部信号の同期をとるために使用。システム・クロックを接続する
\bar{MI}	入力	CPUの \bar{MI} 信号を接続する。 \bar{IORQ} 信号とともにインターラプト・アクノレッジ信号として使用したり、 \bar{RETI} 命令を解釈するのに用いる
\bar{IORQ}	入力	CPUの \bar{IORQ} 信号に接続する。CPUとSIO間でデータを転送する場合、B/ \bar{A} 、C/ \bar{D} 、 \bar{CD} 、および \bar{RD} と組み合わせる使用
\bar{RD}	入力	CPUの \bar{RD} 信号と接続する。 \bar{CE} と \bar{IORQ} がアクティブで、かつ \bar{RD} もアクティブであればSIOからデータを読み出す。 \bar{CE} と \bar{IORQ} がアクティブで、 \bar{RD} が非アクティブであればSIOにデータを書き込む
\bar{RESET}	入力	SIOをリセット
IEI	入力	割り込みの優先順位を決めるデジィ・チェーン回路をIEOと共に構成する。この信号が“H”の時、SIOは割り込みを行うことができる
IEO	出力	IEIと共にデジィ・チェーン回路を構成する。IEOが“L”の場合、そのデバイスあるいはそれより優先順位の高いデバイスが割り込みを実行中であることを示す
\bar{INT}	出力	SIOが割り込み要求をすると“L”になる
W/ \bar{RDY}_A W/ \bar{RDY}_B	出力	DMAコントロール用レディ線、またはCPUとSIO間のデータ転送速度を合わせるためのウェイト線として利用
\bar{CTS}_A \bar{CTS}_B	入力	オート・イネーブルにSIOがプログラムされている場合、この信号をアクティブにすると対応するトランスミッタがイネーブルになる
\bar{DCD}_A \bar{DCD}_B	入力	オート・イネーブルにSIOがプログラムされている場合、この信号をアクティブにすると対応するレシーバがイネーブルになる
RxD _A RxD _B	入力	受信データ線
TxD _A TxD _B	出力	送信データ線
RxC _A RxC _B	入力	受信クロックで、受信データは \bar{RxC} の立ち上がりでサンプリングされる
\bar{TxC}_A \bar{TxC}_B	入力	送信クロックで、送信データは \bar{TxC} の立ち下がりエッジで変化する
RTS _A RTS _B	出力	書き込みレジスタ内のRTSビットをセットするとアクティブになる。SIOが送信したいときにRTSビットをセットする
\bar{DTR}_A \bar{DTR}_B	出力	書き込みレジスタ内のDTRビットをセットするとアクティブになる。SIOが受信できる状態になればこの信号をアクティブにする
SYNC _A SYNC _B	双方向	外部同期通信に使用する場合、外部同期が成立したらこの信号をアクティブする

図12にZ80 SIOのピン配置を示しますが、チャンネルAにはすべての信号があります。表2に信号の説明を、図13に内部構造を示します。

■ Z80 SIOのレジスタの設定

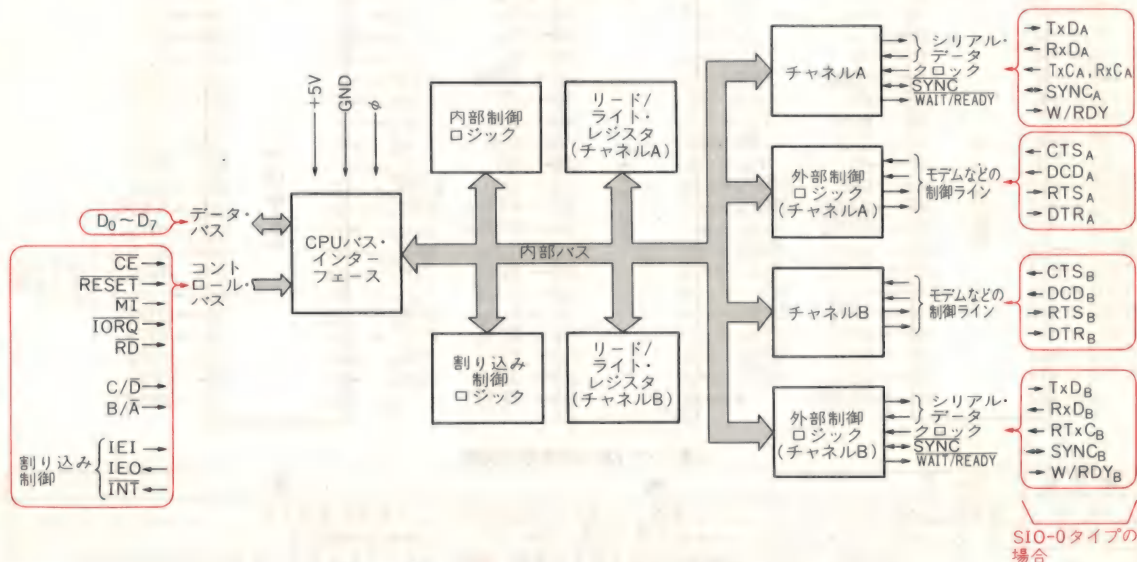
図14に内部レジスタを示します。レジスタの設定は、まずWR₀で設定するレジスタを選択したうえでレジ

スタに書き込みます。WR₀のときは、1回です。

このとき、B/A信号で設定するチャンネルを選択し、C/D信号は“1”にします。

WR₀は、レジスタの選択とコントローラの動作を指定します。WR₁は、割り込みコントロール・レジスタで、割り込みを使用しない場合は、すべて“0”にします。WR₂もWR₁と同様で割り込み時のジャン

〈図13〉^(a) Z80 SIOの内部構造



〈図14〉⁽¹¹⁾ Z80 SIOのレジスタ

書き込みレジスタ0: WR₀

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀		A	B	S
0	0	0	レジスタ0	レジスタ・ポインタ	0	0	0	レジスタ0	○	○	○
0	0	1	レジスタ1		0	0	1	レジスタ1	○	○	○
0	1	0	レジスタ2		0	1	0	レジスタ2	○	○	○
0	1	1	レジスタ3		0	1	1	レジスタ3	○	○	○
1	0	0	レジスタ4		1	0	0	レジスタ4	○	○	○
1	0	1	レジスタ5		1	0	1	レジスタ5	○	○	○
1	1	0	レジスタ6		1	1	0	レジスタ6	○	○	○
1	1	1	レジスタ7		1	1	1	レジスタ7	○	○	○
0	0	0	動作に何の影響も与えない	制御コマンド	0	0	0	動作に何の影響も与えない	○	○	○
0	0	1	アボート送出		0	0	1	アボート送出	○	○	○
0	1	0	外部ステータス割り込みリセット		0	1	0	外部ステータス割り込みリセット	○	○	○
0	1	1	チャンネル・リセット		0	1	1	チャンネル・リセット	○	○	○
1	0	0	つぎの受信キャラクタで割り込みイネーブル		1	0	0	つぎの受信キャラクタで割り込みイネーブル	○	○	○
1	0	1	送信割り込みの保留リセット		1	0	1	送信割り込みの保留リセット	○	○	○
1	1	0	エラー・リセット		1	1	0	エラー・リセット	○	○	○
1	1	1	割り込みからの復帰(Aチャンネルのみ)		1	1	1	割り込みからの復帰(Aチャンネルのみ)	○	○	○
0	0	0	動作に何の影響も与えない	CRCリセット・コード	0	0	0	動作に何の影響も与えない	○	○	○
0	1	0	受信CRCチェッカ・リセット		0	1	0	受信CRCチェッカ・リセット	○	○	○
1	0	0	送信CRCジェネレータ・リセット		1	0	0	送信CRCジェネレータ・リセット	○	○	○
1	1	0	送信アンダラン/EOMリセット		1	1	0	送信アンダラン/EOMリセット	○	○	○

書き込みレジスタ1: WR₁

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀		A	B	S
0	0	0	外部/ステータス割り込みイネーブル	割り込みモード	0	0	0	外部/ステータス割り込みイネーブル	○	○	○
0	0	1	送信割り込みイネーブル		0	0	1	送信割り込みイネーブル	○	○	○
0	1	0	ステータス・エフェクト・ベクタ・イネーブル(Bチャンネルのみ)		0	1	0	ステータス・エフェクト・ベクタ・イネーブル(Bチャンネルのみ)	○	○	○
0	1	1	ステータス・エフェクト・ベクタ・イネーブル(Bチャンネルのみ)		0	1	1	ステータス・エフェクト・ベクタ・イネーブル(Bチャンネルのみ)	○	○	○
0	0	0	受信割り込みディセーブル	ウェイト・レディ機能選択	0	0	0	受信割り込みディセーブル	○	○	○
0	0	1	最初のキャラクタのみで受信割り込み		0	0	1	最初のキャラクタのみで受信割り込み	○	○	○
0	1	0	すべての受信キャラクタで割り込み(パリティ・エラーは特別受信条件となる)		0	1	0	すべての受信キャラクタで割り込み(パリティ・エラーは特別受信条件となる)	○	○	○
0	1	1	すべての受信キャラクタで割り込み(パリティ・エラーは特別受信条件とならない)		0	1	1	すべての受信キャラクタで割り込み(パリティ・エラーは特別受信条件とならない)	○	○	○
1	0	0	WAIT : フローティング	ウェイト・レディ機能選択	1	0	0	WAIT : フローティング	○	○	○
1	0	1	READY : "H"レベル		1	0	1	READY : "H"レベル	○	○	○
1	1	0	WAIT : 送信バッファがフルかつSIOのデータ・ポートが選択されているとき"L"レベル : 送信バッファが空のときフローティング		1	1	0	WAIT : 送信バッファがフルかつSIOのデータ・ポートが選択されているとき"L"レベル : 送信バッファが空のときフローティング	○	○	○
1	1	1	READY : 送信バッファがフルのとき"H"レベル : 送信バッファが空のとき"L"レベル		1	1	1	READY : 送信バッファがフルのとき"H"レベル : 送信バッファが空のとき"L"レベル	○	○	○
0	0	0	WAIT : 受信バッファがフルのときフローティング : 受信バッファが空かつSIOのデータ・ポートが選択されているとき"L"レベル		0	0	0	WAIT : 受信バッファがフルのときフローティング : 受信バッファが空かつSIOのデータ・ポートが選択されているとき"L"レベル	○	○	○
0	0	1	READY : 受信バッファがフルのとき"H"レベル : 受信バッファが空のとき"L"レベル		0	0	1	READY : 受信バッファがフルのとき"H"レベル : 受信バッファが空のとき"L"レベル	○	○	○
0	1	0	WAIT : 送信バッファがフルかつSIOのデータ・ポートが選択されているとき"L"レベル : 送信バッファが空のときフローティング		0	1	0	WAIT : 送信バッファがフルかつSIOのデータ・ポートが選択されているとき"L"レベル : 送信バッファが空のときフローティング	○	○	○
0	1	1	READY : 送信バッファがフルのとき"H"レベル : 送信バッファが空のとき"L"レベル		0	1	1	READY : 送信バッファがフルのとき"H"レベル : 送信バッファが空のとき"L"レベル	○	○	○

(注) A: 非同期, B: 同期, S: SDLC の各通信方式にて, /: 未使用(0にプログラム), ○: 使用を意味する。

〈図14〉⁽¹¹⁾ Z80 SIOのレジスタ(つづき)

書き込みレジスタ2: WR₂

D₇ D₆ D₅ D₄ D₃ D₂ D₁ D₀

V₀
V₁
V₂
V₃
V₄
V₅
V₆
V₇

割り込みベクタ
(Bチャネルのみ)

書き込みレジスタ4: WR₄

D₇ D₆ D₅ D₄ D₃ D₂ D₁ D₀

		A	B	S
パリティ・イネーブル		○	○	／
パリティ偶数(1)奇数(0)選択		○	○	／
モード選択		／	／	／
ビット/キャラクタ	非同期 モード 選択	／	○	○
ストップビット/キャラクタ		○	／	／
ビット/キャラクタ		／	／	／
同期モード	同期モード	／	○	○
同期モード		／	／	／
同期モード		／	／	○
同期モード		／	○	／
6	クロック・レート	○	／	／
2		○	／	／
4		○	／	／
		○	／	／

書き込みレジスタ6: WR₆

D₇ D₆ D₅ D₄ D₃ D₂ D₁ D₀

	A	B	S
同期キャラクタ・ビット0	○	○	○
同期キャラクタ・ビット1	○	○	○
同期キャラクタ・ビット2	○	○	○
同期キャラクタ・ビット3	○	○	○
同期キャラクタ・ビット4	○	○	○
同期キャラクタ・ビット5	○	○	○
同期キャラクタ・ビット6	○	○	○
同期キャラクタ・ビット7	○	○	○

読み出しレジスタ0: RR₀

D₇ D₆ D₅ D₄ D₃ D₂ D₁ D₀

	A	B	S
受信キャラクタ有効	○	○	○
割り込み保留(Aチャネルのみ)	○	○	○
送信バッファ空	○	○	○
DCD	○	○	○
シンク/ハント	○	○	○
CTS	○	○	○
送信アンダラン/EOM	○	○	○
ブレーク/アポート	○	○	○

読み出しレジスタ2: RR₂

D₇ D₆ D₅ D₄ D₃ D₂ D₁ D₀

V ₀	ステータス・ アフェクト・ ベクタがセッ トされている 場合変化 する	ベクタ (Bチャネルのみ)
V ₁		
V ₂		
V ₃		
V ₄		
V ₅		
V ₆		
V ₇		

書き込みレジスタ3: WR₃

D₇ D₆ D₅ D₄ D₃ D₂ D₁ D₀

	A	B	S
受信イネーブル	○	○	○
同期キャラクタ・ロード禁止	○	○	○
アドレス・サーチ・モード	○	○	○
受信CRCイネーブル	○	○	○
エンタ・ハント・フェーズ	○	○	○
オート・イネーブル	○	○	○
0 0 5ビット/キャラクタ	○	○	○
0 1 7ビット/キャラクタ	○	○	○
1 0 6ビット/キャラクタ	○	○	○
1 1 8ビット/キャラクタ	○	○	○
		受信ビット/キャラクタ	

書き込みレジスタ5: WR₅

D₇ D₆ D₅ D₄ D₃ D₂ D₁ D₀

		A	B	S
送信CRCイネーブル		○	○	○
送信要求(RTS)		○	○	○
CRC16(1),CRC-CCITT(0)選択		○	○	○
送信イネーブル		○	○	○
ブレイク送出		○	○	○

ラクタ	通信ビット・ キャラクタ	○	○	○
ラクタ		○	○	○
ラクタ		○	○	○
ラクタ		○	○	○

DTR		○	○	○
-----	--	---	---	---

書き込みレジスタ7: WR₇

D₇ D₆ D₅ D₄ D₃ D₂ D₁ D₀

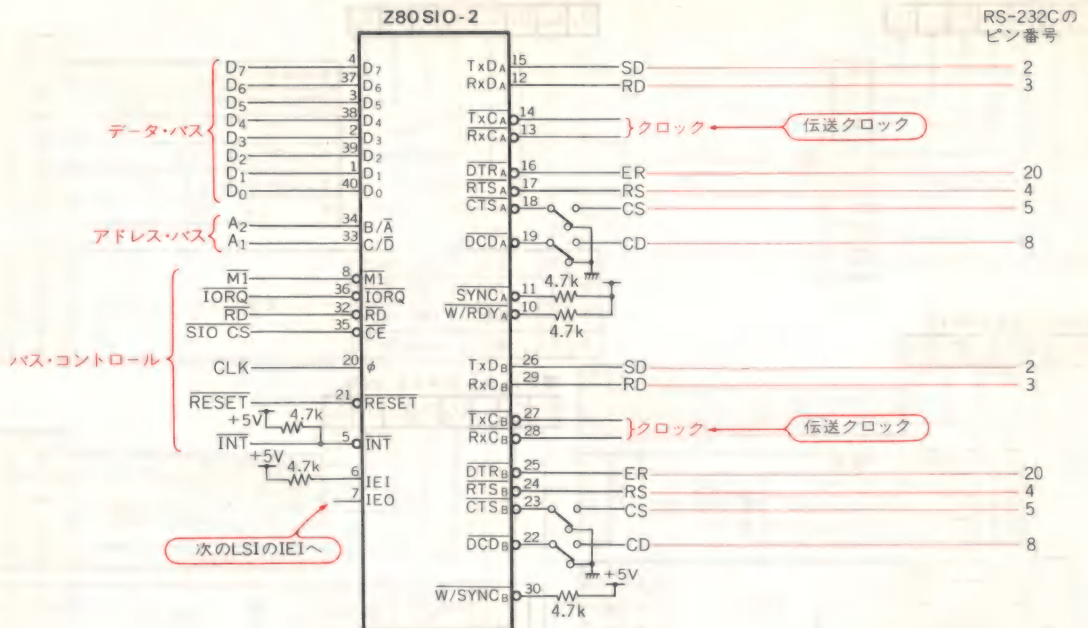
	A	B	S
同期キャラクタ・ビット8	○	○	○
同期キャラクタ・ビット9	○	○	○
同期キャラクタ・ビット10	○	○	○
同期キャラクタ・ビット11	○	○	○
同期キャラクタ・ビット12	○	○	○
同期キャラクタ・ビット13	○	○	○
同期キャラクタ・ビット14	○	○	○
同期キャラクタ・ビット15	○	○	○

読み出しレジスタ1: RR₁

D₇ D₆ D₅ D₄ D₃ D₂ D₁ D₀

	A	B	S
全キャラクタ送出	○	○	○
1バイトまの 17フィールド・ビット	2バイトまの 17フィールド・ビット		端 数 コ ド
1 0 0 0	0	3	
0 1 0 0	0	4	
1 1 0 0	0	5	
0 0 1 0	0	6	
1 0 1 0	0	7	
0 1 1 0	0	8	
1 1 1 0	1	8	
0 0 0 0	2	8	
パリティ・エラー	○	○	○
受信オーバーラン・エラー	○	○	○
CRC/フレーミング・エラー	○	○	○
エンド・オブ・フレーム	○	○	○

〈図15〉^{(S),(R)} Z80SIOの回路図



プ・ベクトルを指定します。

WR₃は、8251Aのコマンドにあたり、受信の制御を指定します。WR₄は、8251Aのモードにあたりコントローラのモードを指定します。WR₅は、送信の制御を指定します。WR₆, WR₇は、同期モード、SDLCモード時に指定します。

ステータス・レジスタは、RR₀だけで直接読み出すことができます。RR₁, RR₂はWR₀でレジスタを指定します。

RR₀は、データの受信/送信状態がセットされます。RR₁は、エラーの状態とSDLCモードでの受信ビットの端数がセットされます。RR₂は、割り込み時のベクトルがセットされます。

回路例を図15に示します。回路図は、SIO-2のタ

イプを用い2チャンネル使っています。DMAは使用しません。

● Z80 SIOのプログラミング

参考までにSDLCのプログラムをリスト3に示します。初期化ルーチンでは、コントローラをSDLCモードにして、送受信を可能にしています。

送信ルーチンではフレームを開始して、CRCの計算、フレーム終了時にCRCを送ります。受信ルーチンは、フレームを受信してCRCのチェックをします。

8080系用に日本電気製のμPD7201があります。基本機能は、Z80 SIOと同じです。このLSIについては「トランジスタ技術スペシャルNo.8」を参照してください。

〈リスト3〉 Z80 SIOによるシリアル入出力(Lattice C)

```
#define chadat 0x00 /* チャンネル A データ */
#define chacmd 0x01 /* チャンネル A コマンド */
#define chasts 0x01 /* チャンネル A ステータス */

#define reset 0x18 /* リセット コマンド */
#define mmode 0x20 /* SDLC モード */
#define rmode 0xcc /* 受信 モード */
/* キャラクタ 8 ビット */
/* CRC 許可、アドレス・サーチ */
#define smode 0xe3 /* 送信 モード */
/* DTR オン、RTS オン、CRC 許可 */
/* キャラクタ 8 ビット */
/* チェック アドレス */
/* フラグ シーケンス */
#define caddr 0x01 /* 受信レディ・ビット */
#define sflag 0x7e /* 送信レディ・ビット */
#define rrdy 0x01 /* エラー、エンド・ビット */
#define srdy 0x04
#define err 0xf0
```


〈リスト3〉 Z80 SIOによるシリアル入出力(つづき)

```

/* コントローラ 初期化 */
sio_init()
{
    outp(chacmd,reset);      /* reset */

    outp(chacmd,0x01);      /* wr1 select */
    outp(chacmd,0x00);      /* intr disable */

    outp(chacmd,0x04);      /* wr4 select */
    outp(chacmd,mmode);     /* SDL mode set */

    outp(chacmd,0x03);      /* wr3 select */
    outp(chacmd,rmode);     /* recive mode set */

    outp(chacmd,0x05);      /* wr5 select */
    outp(chacmd,smode);     /* send mode set */
    outp(chacmd,0x06);      /* wr6 select */
    outp(chacmd,caddr);     /* check address */

    outp(chacmd,0x07);      /* wr7 select */
    outp(chacmd,sflag);     /* flag bit */
    return(0);
}

/* 受信 ルーチン */
recv(data)
char data;
{
    int stat,er_cd;

    outp(chacmd,0x43);      /* wr3 select and crc reset */
    outp(chacmd,rmode+0x11); /* auto hant mode */

    while(((stat=inp(chasts))&rrdy)!=rrdy) /* 受信待ち */
    {
        if((stat&0x80)!=0) /* アボート? */
        {
            return(-2); /* アボート終了 */
        }
        outp(chacmd,0x01); /* rr1 select */
        if(er_cd=(inp(chasts)&err)!=0) /* r r 1 リード */
        {
            outp(chacmd,0x30); /* エラー・リセット */
            return(er_cd); /* エラー、エンド・フレーム */
        }
    }

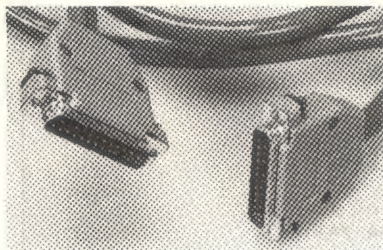
    /* データ 受信 */

    data=inp(chadat);
    return(0);
}

/* データ 送信 */
send(data)
char data;
{
    outp(chacmd,0x85);      /* wr5 select and crc reset */
    outp(chacmd,smode+0x04); /* 送信許可 */
    outp(chacmd,0xc0);      /* reset underrun */

    while((inp(chasts)&srady)==0) /* 送信レディ? */
    {}
    outp(chadat,data);      /* データ送信 */
    return(0);
}

```



§ 2-3

PC9801 による BSC 伝送 の実際

沖野 新

伝送制御手順

ここでは、BSC(バイシンク)によるデータ伝送を行う手順を、パソコンを用いて説明していきます。

BSCでは、図1のシーケンスにしたがって伝送を行います。つまり、送信側と受信側がそれぞれ相互に結ばれたことを確認し、その後テキストを送ります。そして、通信が終わったので、回線を切るといった手順を決めてあるわけです。

回線接続シーケンス

● 送信側

ターミナル呼び出しを行うためにENQを送信します。ターミナルが複数台あれば、IDにターミナル番

号を付加します。その後、受信状態に入り応答受信待ちになります。

ACK₀を受信した場合は、正常に回線接続が行われたと見なし、テキストの送信を行います。

NAKを受信した場合は、再度ENQを送信します。

受信側の応答がない場合は、通常約2～5秒時間監視を行い、時間監視終了後(以下タイム・アウトと呼ぶ)、ENQの再送を行います。

WACK, ACK₁を受信した場合は、NAKと同様の処理を行います。

● 受信側

ENQ受信後、以後テキスト・ブロックを正常に受信できる状態であれば、ACK₀を送信します。正常に受信できる状態でなければ、WACKを送信します。

それ以外の制御コードであれば、応答は行いません。

伝送コードについて

図Aに、伝送コードを示します。各レコード先頭には、PAD_Lが1バイト、SYNが1または2バイトあります。これは、BSC手順が同期式で伝送されるからです。これによって、伝送キャラクタの同期を行います。

PAD_Lコードは、SYNコードより先行して、同期キャラクタの位置合わせに使用されます。

PAD_TはF F hで、レコードの最後を示します。

● ENQコード

このコードは送信側から送られます。これによって、受信側はデータ受信の状態に入ります。IDの部分は、複数のターミナルの選択に使用されます。1台であれば不要です。

● STXコード

このコードはテキスト・レコードであることを示します。STXコードの次から、伝送するテキストを送ります。通常テキスト長は256バイトです。このレコードの後には、ETBまたはETXとBCC(Block Check Character)です。

ETBは、次に送るSTXコードがある場合に使用します。

ETXは、最後のSTXコードであることを示します。

BCCは、テキスト部のチェック・キャラクタです。これについては別に述べます(p.58参照)。

● EOTコード

このコードは送信側の終了を示します。これによって伝送を終結させます。

● ACK₀およびACK₁

これは送信データを正常に受信したときに、受信側から送るコードです。

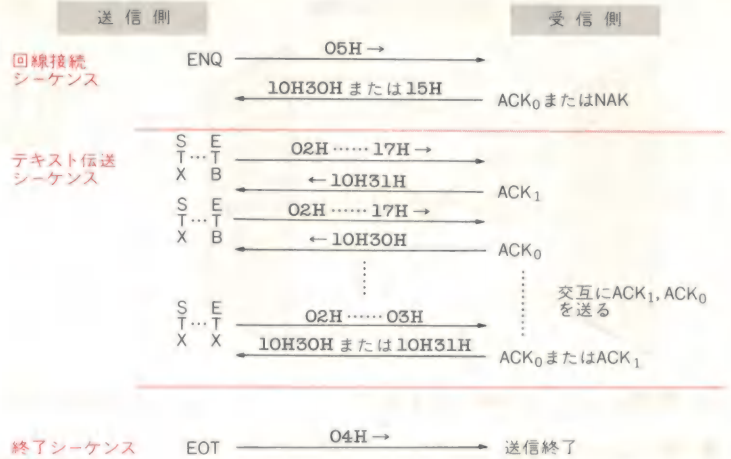
ENQコードを受信した場合は、ACK₀を送信します。以後、STXコード受信ごとに、交互にACK₁, ACK₀を送信します。

● NAKコード

このコードは、受信側が正常にデータを受信できないときに送信します。または、BCCエラー、シーケンス・エラーのときにも送ります。

送信側は、これによってテキスト・レコードの再送信、または再度ENQコードから始めます。

〈図1〉
伝送制御シーケンス



● テキスト伝送シーケンス

● 送信側

回線接続後、テキスト・ブロックの送信を行います。
ACK₀, ACK₁を受信した場合には、テキストがあればテキストを送信し、なければ終了信号のEOTを送信します。

テキスト・ブロックは、STXから始まり、ETB, ETXまでです。ETBは、これに連続するテキスト・ブロックがある場合に使用し、ない場合にはETXを使用します。

NAKを受信した場合は、直前のテキスト・ブロックを再度送信します。

タイム・アウトであれば、再度回線接続(ENQ)から始めます。

WACKを受信した場合は、ただちにENQの送信を行います。

● 受信側

テキスト・ブロックの受信を行います。

テキストを正常に受信し、BCCエラーがない場合に、ACK₀またはACK₁を送信します。

正常に受信できない、もしくはBCCエラーが発生した場合には、NAKを送信します。

ENQを受信した場合は、直前に送信したACK₀, ACK₁を再度送信します。

EOTを受信した場合は、データ伝送の終了と見な

● WACKコード

このコードは受信側が受信状態を一時中断する場合に使います。

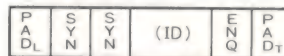
● RVIコード

受信側が送信側になる場合に、このコードを送信側に送ります。

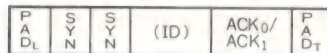
〈図A〉

伝送制御レコード・フォーマット

ENQコード



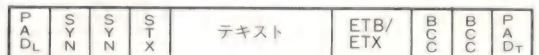
ACK₀
または
ACK₁



NAK



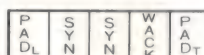
STX
+
ETB/ETX



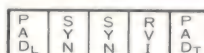
EOT



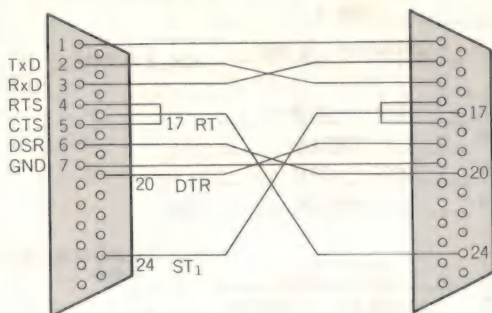
WACK



RVI



〈図2〉 同期式RS-232Cの接続



し、受信シーケンスを終了させます。

● 終了シーケンス

● 送信側

テキスト・ブロックをすべて送信した場合に、EOTを送信します。

伝送中のタイム・アウト・エラーが発生した場合にも、EOTを送信します。

● 受信側

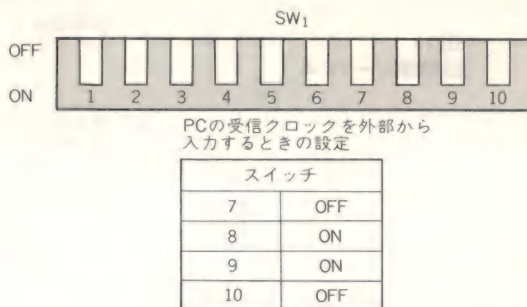
すべての受信処理を終了させます。

次に実際パソコン上で動くプログラムを示します。

ハードウェアの構成

最近のパーソナル・コンピュータは、ほとんどRS-

〈図3〉 PC9801のディップ・スイッチの設定



232Cが標準となっていますので、容易にBSC伝送が行えます。

PC9801を用いて、BSC伝送を行った例をここで説明します。ほかのパーソナル・コンピュータも同様に使用できます。

PC9801(以下PCとする)は、RS-232C用のコントローラに8251Aを使用しています。

BSCでは、前にも述べたように、同期式で通信しなければなりませんが、PCでは非同期式が標準となっていますから、同期式に変更します。これは、直接コントローラに対して、モード設定を行うことでできます。

同時に、伝送速度クロックも変更する必要があります。PCでは8253を使っています。このプログラムでは4800bpsにしました。

BCC(Block check character)とは

伝送中にテキストのデータ抜けまたは、パリティでは発見不可能な二重パリティ・エラーなどを容易に発見するための2バイトのキャラクタです。

BSCの伝送では、CRC16またはCRC12が使用されています。

最近のコントローラでは、このCRCの計算ロジックを内蔵していますが、8251Aではソフトウェア

によって実現できます。

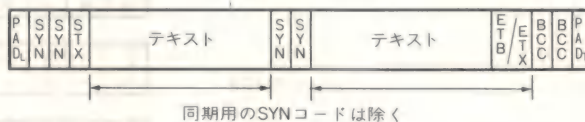
BCCのチェック範囲は、STXコードの次のテキストからETBまたはETXまでです。

チェックの方法は、送信側ではBCCを計算し、その結果を送信します。受信側は、データの受信を行いながらBCCを計算して、1レコード受信後、BCCの比較を行います(図B参照)。

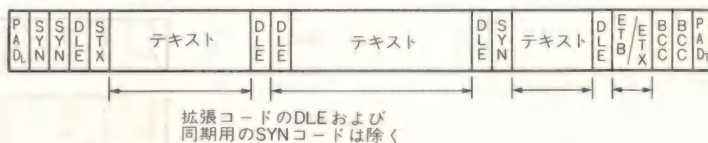
〈図B〉

BCCの計算範囲

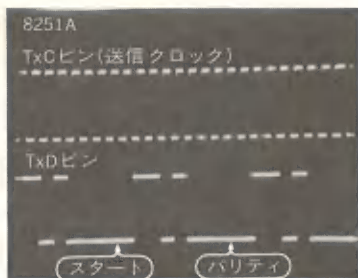
非透過モード



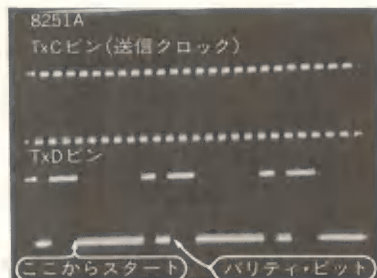
透過モード



〈写真1〉
SYNコードの送信
[X: 2 ms/div,
Y: 2 V/div]



〈写真2〉
“A”の送信
[X: 2 ms/div,
Y: 2 V/div]



RS-232Cの接続は、無手順と同様ですが、同期式の場合は、相手の送信クロックを受信クロックとしたほうがよいでしょう。

図2に接続方法、図3にPCでの送受信クロックの変更方法を示します。

参考として同期式伝送時の波形を写真1、写真2に示します。共に伝送速度は、4800bpsで8251AのTxDの状態です。

写真1は、SYNコード(16H)を送信したときの波形です。写真2は、“A”(41H, 01000001B)を送信したときの波形です。

伝送プログラム

次に、パーソナル・コンピュータを使った、ファイル伝送について必要なことがらを説明します。

●初期化ルーチン

このルーチンでは、インターフェース・コントローラ8251Aおよび伝送速度の設定を行います。

モード設定は、同期モード、キャラクタ長は8ビット、奇数パリティ、同期キャラクタは2バイトでSYN(32H)とします。

●受信メイン・ルーチン

伝送レコードを受信して、各処理を行います。

●送信メイン・ルーチン

回線をオープンし、テキスト・ブロックを伝送します。

●受信処理

ハント・モードを設定後、ハント状態に入ります。ハントは、同期キャラクタを受信後、ステータス・フラグをセットすることによって終わります。同期確立後、キャラクタの受信を行います。レコードの終了は、PAD_rを受信後です。

STXを受信した場合は、ETBまたはETXまでをテキストとして受信します。そしてこれに続くBCC 2バイトを受信しBCCの比較を行います。

受信バッファ長は、1テキスト長256バイトとして最大300バイト確保します。

好評発売中

定本 続トランジスタ回路の設計

●FET/パワーMOS/スイッチング回路を実験で解析

増幅回路以外にも広く使われているトランジスタ。さらにパワーMOS FETの台頭により応用分野が広がってきたFET、FET増幅回路の基礎実験からはじまり、スイッチング回路から発振回路までをやさしく実験で解説しています。

鈴木雅臣 著

A 5判 360頁

定価2,752円(税込)

定本 トランジスタ回路の設計

●増幅回路技術を実験を通してやさしく解析

本書は多忙な技術者、あるいは技術者をめざす人のために用意した、とてもとてもわかりやすいトランジスタ回路の本です。本書は大好評であったトラ技ORIGINAL No.1とNo.5の中からトランジスタ増幅回路について精選し、さらに大幅に加筆を行った、たぶん最後のトランジスタ回路の解説書です。

鈴木雅臣 著

A 5判 324頁

定価2,243円(税込)

定本 OPアンプ回路の設計

●再現性を重視した設計の基礎から応用まで

本書は14章で構成されており、第1章から第5章までは基礎的な技術を、第6章から第14章までは具体的な各種の応用技術を解説しています。

岡村迪夫 著

A 5判 424頁

定価2,854円(税込)

BSC伝送のクロック

PC9801VMでは、8251Aのクロックの切り替えを変更しているために、受送信共にクロックを外部から入力しなければなりません。このため、各接続コネクタのST₁(24)ピンとST₂(15)ピンを接続する必要があります。

ディップ・スイッチは、5番目をONにします(図C参照)。

〈図C〉⁽⁹⁾ PC9801のSW₁の5/6の機能

SW ₁	5	RS-232Cの伝送速度(ボーレート)を決めるためのタイム選択	スイッチ5	スイッチ6	
	6				
			ON	ON	BCI同期
			ON	OFF	ST ₂ 同期
			OFF	ON	同期刻時機構
			OFF	OFF	調歩同期(非同期)

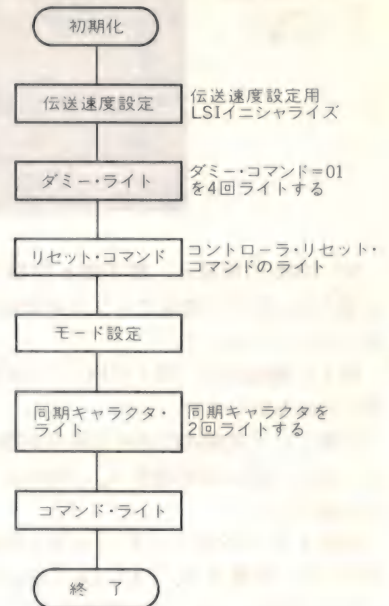
備考

スイッチ5, 6はRS-232Cの伝送速度を決めるためのタイムを選択する。

VMのSW₁の6は未使用。

スイッチ5	スイッチ6	機能
ON	ON	BCI同期……送信用のタイミングとして、PC9801VXの内部タイムを使用する。受信用のタイミングはモデムより供給されるクロックを使用する。
ON	OFF	ST ₂ 同期……送、受信用のタイミングとしてモデムより供給されるクロックを使用する。

〈図4〉同期式初期化フローチャート



受信エラー時とタイム・アウトの場合は、すぐに受信を中断します。このプログラムでは、約2秒のタイム・アウトに設定しています。デバッグ時には、長くしたほうがよいと思われます。

● 送信処理

各伝送制御コードにPAD_LおよびSYNを付加して送信します。

テキストを送信する場合、あらかじめ、STX, BCC, ETBまたはETXを付加したレコードを用意しておき、これを送信します。

テキスト長は最大256バイトとします。

8251Aに対してI/Oの操作を行うルーチンは、無手順の場合と同様です。

これらのフローチャートを、図4～図7に示します。プログラムをリスト1に示します。このプログラムは、MS-DOS上に作成しました。

● おわりに

最後に、BSC手順は同期式のプロトコルですが、非同期式でも使用できます。この場合、同期キャラクタおよび同期を確立するハント・モードが不要となります。

しかし、スタート/ストップ・ビットが付加されますから、伝送効率が約20%ほど悪くなります。

また、割り込み処理を使用しない場合に、伝送中、ディスクなどをアクセスすると、受信側でタイム・アウトが発生することがあります。

トランジスタ技術 No.6 SPECIAL

B5判 168頁 定価1,570円(税込)

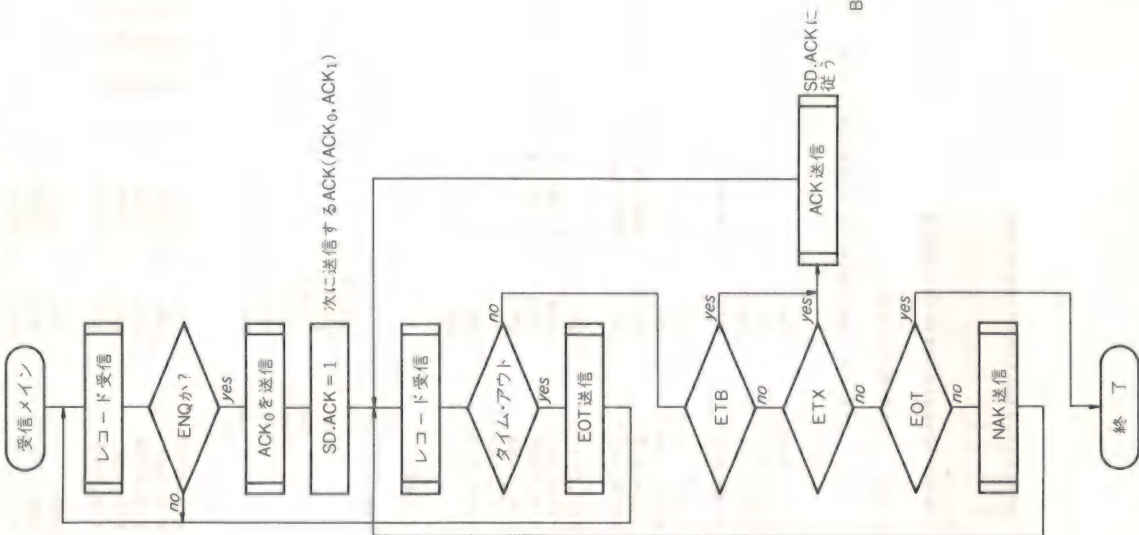
現在一番使われているZ80とその周辺LSIをとりあげ、割り込み技術、マクロ命令の使い方まで、詳細に説明します。

基礎からマクロ命令を使いこなすまでのノウハウを集大成
Z80ソフト&ハードのすべて

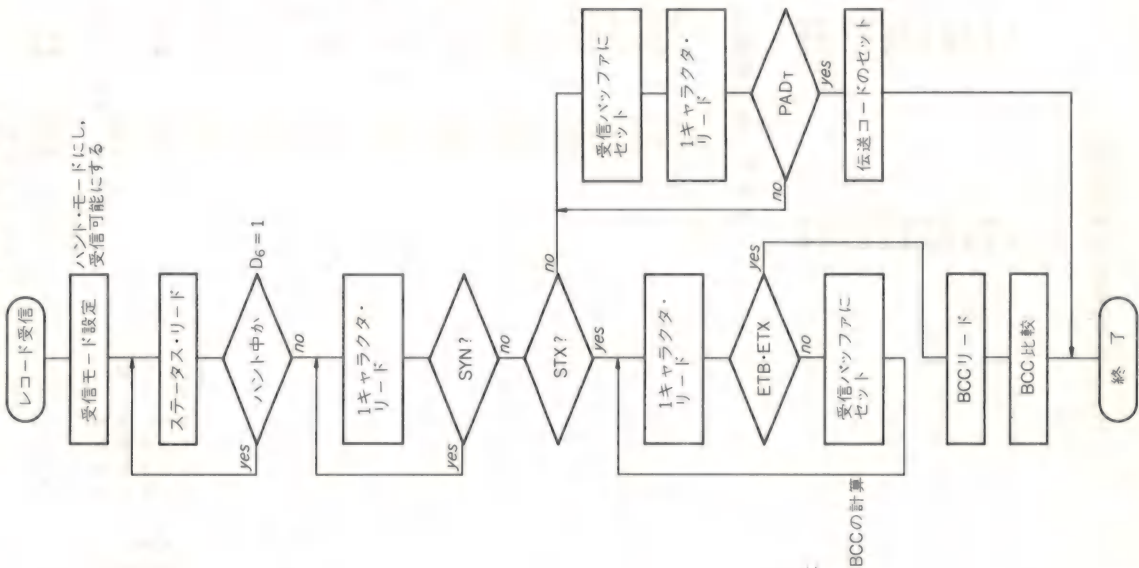
● 目次 ●

マイコン・システムの基本構成/アセンブラの基礎/システム構成の基本/メモリとの接続/パラレル・インターフェース/シリアル・インターフェース/カウンタ/タイマの使い方/割り込みのプログラミング/上級プログラミング/8048クロス・アセンブラ

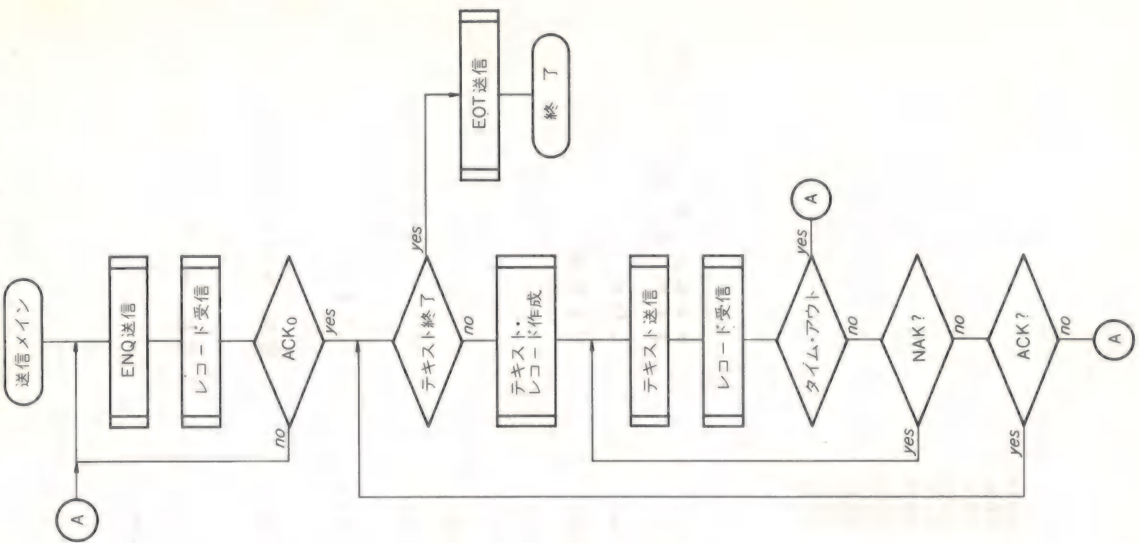
〈図5〉受信メイン・フローチャート



〈図6〉レコード受信フローチャート



〈図7〉送信メイン・フローチャート



001h	equ	001h	equ
003h	equ	003h	equ
026h	equ	026h	equ
03dh	equ	03dh	equ
010h	equ	010h	equ
037h	equ	037h	equ
06bh	equ	06bh	equ
070h	equ	070h	equ
061h	equ	061h	equ
055h	equ	055h	equ
0ffh	equ	0ffh	equ

```
assume cs:code,ds:code,es:code
```

8251A 初期化

```

mov     al,0b6h
out     tim_cmd,al
mov     al,0
out     tim_ch2,al
mov     al,2
out     tim_ch2,al
mov     al,1
out     io_cmd,al
call    wait
out     io_cmd,al
call    wait
out     io_cmd,al

```

```

mov     ax,cs
mov     es,ax
mov     ds,ax
init
call    send-e
call    start

```

io-estat	equ	032h	:	8 2 5 1	A	ステータス
io-ond	equ	032h	:	8 2 5 1	A	コマンド
stio-data	equ	030h	:	8 2 5 1	A	ステータス
tim-ch2	equ	075h	:	8 2 5 3	A	チャンネル
tim-ond	equ	077h	:	8 2 5 3	A	コマンド

; ダニークエイト
wait:

push	ax
pop	ax
ret	

モーター
モーター

3

: LOW

: H I G H

イ・エ・フ

ドマコトセリ:

トセーモ 同 様

同編 第六卷 第六號

BSC受信ルーチン

63

トランジスタ技術
SPECIAL


```

cmp      al,ah
je       send_loop
xor      ack_flag,1
jmp      send_entry

send_exit:
call     send_eot
ret

: 'ENQ'
send_enq:
call     send_head
mov      si,offset enq_id
call     send_msg
ret

: 送信 データ バッファ
send_text:
call     send_head
mov      si,offset send_buff
mov      textl:

lodsb
call     al,padt
cmp      jne textl
ret

: 'EOT'
send_eot:
call     send_head
mov      si,offset eot_id
call     send_msg
ret

: ヘッダ部 送信
send_head:
mov      al,23h
out
mov      si,offset head_id
call     send_msg
ret

: 'ACK'
send_ack:
cmp      je send_ack1
: 'ACK0'
send_ack0:
call     send_head
mov      si,offset ack0_id
call     send_msg
ret

: 'ACK1'
send_ack1:
mov      al,ah
push     cx
mov      cx,0

al,sio_stat
al,80h
term_err
al,1
send2
send1
cx

: タイムアウト セット
: ステータス リード ?
: テーミナル エラー ?
: yes ok?
: yes
: 送信 待ち
: 送信 エラー

```

```

send2:      pop      cx
            mov      al,ah
            out      sio_data,al
            cld
            ret

;   コン트롤ー    文字   テーブル
head_id     db      3
            db      pad1
            db      syn,syn
            db      2
            db      enq
            db      pad2
            db      2
            db      eot
            db      pad3
            db      3
            db      die
            db      ack0
            db      pad3
            db      3
            db      die
            db      ack1
            db      pad3
            db      3
            db      die
            db      wack
            db      pad3

;   text_save:
            push     di
            les      si,offset recv_buff
            mov      cx,recv_len
            add      text_len,cx
            rep      movsb
            mov      text_off,di
            es

;   text_get:
            len_get  di,offset send_buff
            mov      al,sta

            ; データ   セット
            ;   送信
            ;
            ;   ユーザ   パッファ   セット
            ;   送信   レングス
            ;   ユーザ   パッファ
            ;   ユーザ   パッファ   に転送
            ;   次のパッファ
            ;   送信   レングス   (CX)
            ;   送信   パッファ
            ;   al,sta

            mov      cx,256
            cmp      text_len,cx
            jnb      len_end
            mov      len_end,
            sub      text_len,cx
            ret

            code     ends
            and

            start

            stosb
            jcxz    crc_cnt,0
            mov     es
            push    si,text_addr
            les
            mov     al,es:[si]
            mov     [di],al
            call    cal_crc
            inc     di
            loop    text_in
            pop     es
            mov     text_len,0
            cmp     text_end
            je      text_off,si
            mov     al,etb
            mov     text_code:
            mov     send_flag,al
            stosb
            call    cal_crc
            mov     al,crc_high
            stosb
            mov     al,crc_low
            stosb
            mov     al,pad1
            stosb
            ret
            text_end:
            mov     al,etx
            jmp     text_code

            ;   CX: 送信レングス
            len_get:
            mov     cx,256
            cmp      text_len,cx
            jnb      len_end
            mov      len_end,
            sub      text_len,cx

            ;   256 以上?
            ;   yes

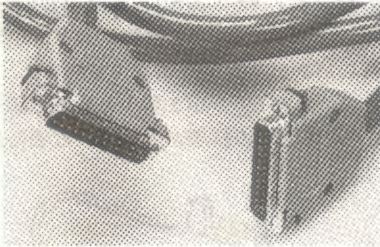
            ;   'STX', 付加
            ;   データ   なし
            ;   CRC   エリア   クリア
            ;   ユーザ   パッファ   セット
            ;   送信   パッファ   に転送
            ;   CRC   計算
            ;   ユーザ   テキスト   終了?
            ;   yes
            ;   'ETB',   セット
            ;   送信   コード
            ;   CRC   計算
            ;   CRC1   セット
            ;   CRC2   セット
            ;   'PADT',   セット
            ;   'ETX',   セット

```


§ 2-4

RS-232C チェッカの製作

鶴野和孝



最近ではほとんどのパソコンにシリアル・インターフェースとしてRS-232Cが標準装備されるようになりました(図1,表1参照)。ところが、これに周辺装置を接続する場合、カタログどおりの組み合わせ(いわゆる純正)ならまず問題はありませんが、そうでない場合にはうまく接続できないことがあります。

カタログや仕様書を理解しているつもりでも、ちょっとした事柄を見落としていたり、ときには仕様書そのものが不完全だったりします。とくにパソコンのマニュアルは厚いので、見過ごしが多くなります。

トラブルがあった場合、ロジック・アナライザなどの測定器が、気軽に使用できる環境ならよいのですが、多くの人はそうでないと思います。そこで簡単なチェッカまたは治具があれば便利と考え、今回チェッカを製作しました。

● トラブルの原因

機器間が相互にマッチングしなければならない条件として、

- (1) ボーレート
- (2) 信号論理
- (3) 信号レベル
- (4) ストップ・ビット数

(5) パリティ

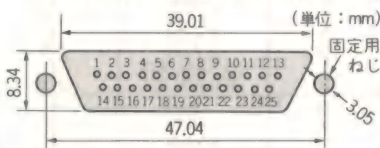
(6) ハンドシェイクのタイミング

(7) ソフト的なデリミタ

などが考えられますが、これらの違いがどのような現象として現れるか考えてみます。

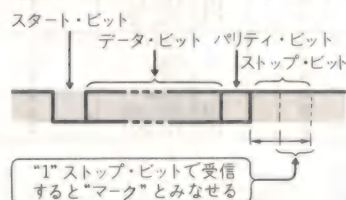
- (1) ボーレートの違いは、フレーミング・エラー、パリティ・エラーおよび送受データの違いとして現れる。
- (2) データ・ラインの論理が逆だと、送信していないときにブレイク・キャラクタとして受信される。ハンドシェイク線(DSR, DTR, RTS, CTSなど)の論理が逆の場合は、送るべきときに送らず、送っていけないときに送るということになってしまう。
- (3) 信号のレベルが合わなければ、まったく動かないとか、悪ければICを壊してしまうこともある。
- (4) ストップ・ビット数がわからないときは“1”で受信するとうまくいく(図2参照)。これは、もし相手が“2”ストップ・ビットで送信しても、二つめのストップ・ビットをマークと見なすから。
- (5) パリティのODD/EVENの違いは、当然パリティ・エラーで、パリティ・ビットの有無の違いはデータ・ビット長に影響し、送受データの違いとして現れたり、パリティ・エラーを発生したりする。

〈図1〉 分界線に使用するコネクタ



(モデム側のコネクタはピン形状がメス型で、接続ケーブル側はオス型とする。
固定用ねじは JIS B 1101 (すりわり付き小ねじ) 丸平小ねじ M2.5 とする)

〈図2〉 シリアル・データのフォーマット



〈表1〉 信号名称表

ピン番号	名称	働き
1	FG	フレーム・グラウンド、筐体のグラウンド
2	TxD	送信データ、端末からモデムへデータを送信する
3	RxD	受信データ、端末はモデムからのデータを受信する
4	RTS	送信要求、端末がデータを送出したい時ONにする
5	CTS	送信許可、端末がデータを送出してよいかどうかをモデムが知らせる
6	DSR	データ・セット・レディ、モデムが送受可能であることを端末に知らせる
7	SG	信号用グラウンド
8	CD	キャリア検出、モデムが回線からキャリアを受信すると、ONを端末につたえる
20	DTR	端末レディ、端末が送受可能であることをモデムに知らせる

(注) このほかのピンも名称や働きが決められているが、ここには一般的なものだけ書き出した

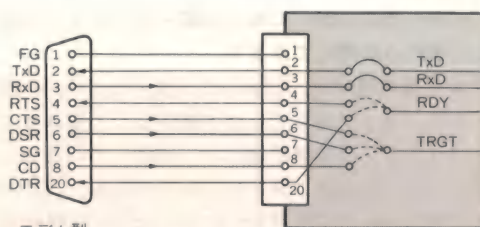
- (6) ハンドシェイクのタイミングが合わないと、データが失われる場合がある。
- (7) デリミタの有無、および食い違いは、それぞれの機器のソフトの働きに依存する。

以上のことから、オシロスコープやテストではわかりにくい事象を拾ってみると、**パリティのチェック**、**ハンドシェイクのタイミング**、**ソフト的なデリミタの有無**の3点になります。これらを重点にチェックする方法を考えたいと思います。

チェッカの操作方法

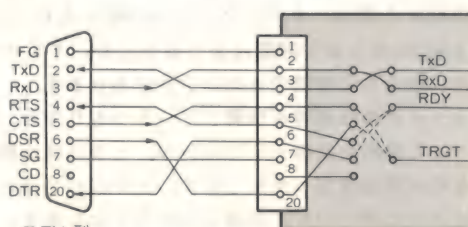
まず相手の信号の入出力の確認のうへ、チェック・

〈図4〉 モデム型と端末型の接続



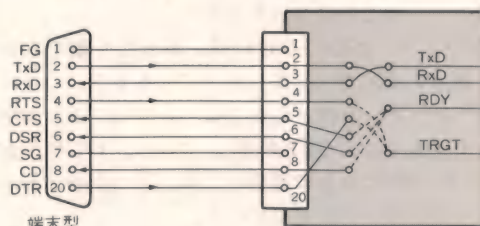
モデム型

(a) ストレート・ケーブル/モデム型の場合



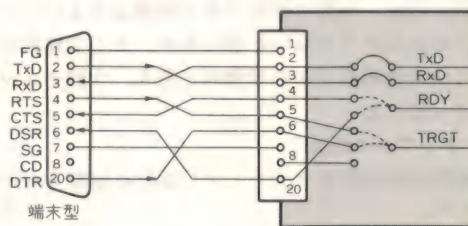
モデム型

(c) クロス・ケーブル/モデム型の場合



端末型

(b) ストレート・ケーブル/端末型の場合



端末型

(d) クロス・ケーブル/端末型の場合

〈表2〉 ディップ・スイッチの設定

▶ DIP SW₁

4	3	2	1	ボーレート (ボー)
ON	ON	OFF	ON	50
ON	ON	OFF	OFF	75
ON	OFF	ON	ON	134.5
ON	OFF	ON	OFF	200
ON	OFF	OFF	ON	600
ON	OFF	OFF	OFF	2400
OFF	ON	ON	ON	9600
OFF	ON	ON	OFF	4800
OFF	ON	OFF	ON	1800
OFF	ON	OFF	OFF	1200
OFF	OFF	ON	ON	2400
OFF	OFF	ON	OFF	300
OFF	OFF	OFF	ON	150
OFF	OFF	OFF	OFF	110

(a) ボーレート表

▶ DIP SW₂

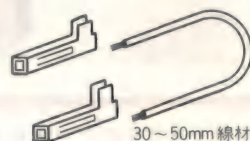
DIP SW					データ・ビット長	パリティ	ストップ・ビット長
5	4	3	2	1			
L	L	L	L	L	5	ODD	1
L	L	L	L	H	5	ODD	1.5
L	L	L	H	H	5	EVEN	1
L	L	L	H	L	5	EVEN	1.5
L	L	H	×	L	5	DISABLED	1
L	L	H	×	H	5	DISABLED	1.5
L	H	L	L	L	6	ODD	1
L	H	L	L	H	6	ODD	2
L	H	L	H	L	6	EVEN	1
L	H	L	H	H	6	EVEN	2
L	H	H	×	L	6	DISABLED	1
L	H	H	×	H	6	DISABLED	2
H	L	L	L	L	7	ODD	1
H	L	L	L	H	7	ODD	2
H	L	L	H	L	7	EVEN	1
H	L	L	H	H	7	EVEN	2
H	L	H	×	L	7	DISABLED	1
H	L	H	×	H	7	DISABLED	2
H	H	L	L	L	8	ODD	1
H	H	L	L	H	8	ODD	2
H	H	L	H	L	8	EVEN	1
H	H	L	H	H	8	EVEN	2
H	H	H	×	L	8	DISABLED	1
H	H	H	×	H	8	DISABLED	2

(注1) × = Don't Care

(注2) "L" = ON, "H" = OFFを示す

(b) モード表

〈図3〉 ショート・ワイヤ



30-50mm 線材

圧着式コネクタ用ソケット・コンタクト

この限りではありません(図4参照)。

表2で示すボーレート表、モード表によりDIP SW₁, DIP SW₂をセットします。これらのディップ・スイッチを変更した場合は、必ずMODE(SW)を押して、モードの再設定を行う必要があります。

● 受信操作

DISPLAY FIFO/RxD(SW)をRxD側に、Rx FREE/STOP(SW)をSTOP側に倒します。

次にターゲット機器から何かデータを送信します。例えば“A B C D E F”, CR, LFを送信したとすると、このチェッカのRx STEP(SW)を押すたびにRB₀~RB₇(LED)に“B”, “D”, “F”, LFと、ひとつおきに表示されます。これは8251AなどのUSARTの特徴のようで、今回使用している機器のシリアル・ポートは、4機種とも外部からビジー状態を送り込んで直ぐには送信停止にはならないで、必ず1バイト余計に送ってきます。

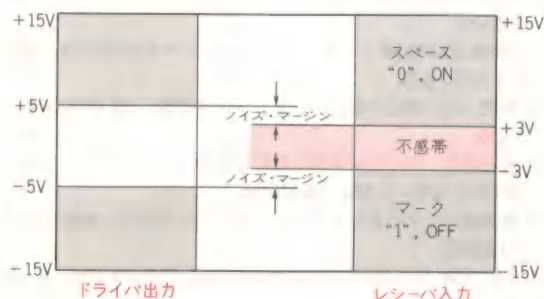
ここで、DISPLAY FIFO/RxD(SW)をFIFO側にする、RB₀~RB₇(LED)には16バイトのFIFOの内容が、Rx STEP(SW)を押すごとに“A”, “B”, “C”, …と、受信時のこのチェッカのビジー状態とともに表示されます。このチェッカのビジーは、READY表示部のRECEIVE(LED)の消灯により表示されます。また、DISPLAY FIFO/RxD(SW)がRxD側にあるときは、ビジー/レディにかかわらず消灯します。下記に述べるSTATUS(LED)も同様です。

また、STATUS IN(PIN)にPE(PIN), OE(PIN), FE(PIN)を接続しておいたならば、同様にデータ受信時のエラー状態をもFIFOに取り込まれ、DISPLAY FIFO/RxD(SW)をFIFO側に倒すと、STATUS(LED)により表示されます。

● 送信操作

ターゲットの機器にデータを送るには、READY表示部のSEND(LED)とTARGET(LED)がともに点灯しているときに、TB₀~TB₇(SW)により送るべきデータを設定し、Tx STEP(SW)を押すことによりデータが送信されます。

〈図5〉RS-232Cのレベル



回路構成

▶ LSI

まず、シリアル通信の中心であるデータのアセンブル/ディスアセンブルはLSIに任せることにします。インターシル社のIM6402を使用するとパリティ・チェックやスタート/ストップ・ビットの検出など、シリアル通信に不可欠な機能が全部LSI任せとなつて手間が省けます。

IM6402はパリティのイネーブル/ディセーブルおよびODD/EVEN、ストップ・ビット数、データ・ビット長の設定が外部よりできるようになっていますので、ここにディップ・スイッチを接続して、任意に変更できるようにします。

▶ ボーレート

ターゲット機器によってボーレートが違いますので、これも変更できるようにしておきます。やはりインターシル社のIM4712/02を使うと、14種類のボーレート・クロックが取り出せます。IM4712を使用すると回路図中のR₁, C₆, C₇は実装しなくてもかまいません。

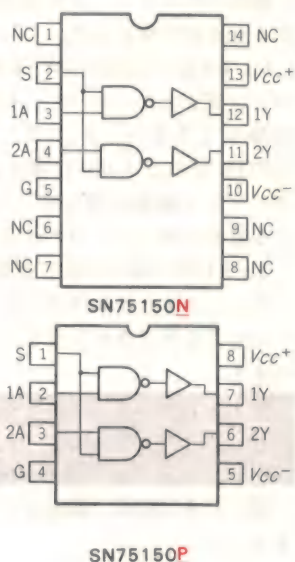
▶ ドライバ/レシーバ

ライン・ドライバにはSN75150を使用します。これはRS-232C用で、±12Vの電源を供給すると規格に合う出力特性が得られます(図5)。このICにはSN75150PとSN75150Nの2種があり、PとNではピン数違います(図6)。しかし中身は同じで、ピン配列も似ていますので、緊急の場合は差し替えることもできます。

▶ チェック・ピン

RS-232Cの信号の使い方には、モデム型と端末型があり、それぞれを接続するのにストレート型、クロ

〈図6〉⁽¹⁾⁽⁵⁾
レベル変換用IC



ス型、変形型などいろいろな接続方法が使われています。これら全部に対応するために、スイッチを使って切り替えていたら大変なことになります。

そこで一番簡単な方法として、Dサブ・コネクタとドライバ/レシーバとの接続はチェック・ピンを立てて、ショート・ワイヤを使用することにしました。ほとんどの場合はTx_D, Rx_D, DTR(またはRTS), DSR(またはCTS)の4本ですみます。また、FIFOのビット数が少ないので、オーバラン・エラー(OE), パリティ・エラー(PE), フレーミング・エラー(FE)もチェック・ピンを立ててショート・ワイヤでSTATUS IN(PIN)に接続します。

▶表示

受信データ(RB₀~RB₇)を点LEDを使ってバイナリ表示します。また、1バイトでもデータが受信されると、必ずFIFOに記憶され、FIFO REMAIN(LED)が点灯します。

RB₀~RB₇(LED)は、DISPLAY FIFO/Rx_D(SW)によりFIFOの出力またはUARTの出力を表示します。

STATUS(LED)はショート・ワイヤで、STATUS IN(PIN)に入力されたPE, OE, FEの状態を表示します(エラーで点灯)。

READY表示としてRECEIVE(LED), SEND(LED)を用意し、このチェッカの状態を表示し、TERGET(LED)でターゲット機器の状態を表します。

▶操作スイッチ

RxSTEP(SW), TxSTEP(SW), Rx FREE/STOP(SW)はチャタリングを考慮し、RSフリップフロップを使います。

DISPLAY FIFO/Rx_D(SW)は上に倒すとFIFOの内容を、下に倒すとUARTの出力をRB₀~RB₇(LED)に表示し、RxSTEP(SW)の出力をターゲット機器へのビジィ、またはFIFOへのリード・パルスとするかを切り替えます。

Rx FREE/STOP(SW)はFREE側(上)に倒すとビジィが禁止され、ターゲット機器はデータをどんどん送信してきます。そこでこのチェッカは、それをFIFOに次々とメモリに蓄え、STOP側(下)に倒すとターゲット機器に対してビジィとなって、“STOP”する直前の16バイトのデータをFIFOに残します。

TxSTEP(SW)の出力は微分し、UARTの送信パルスとします。これはほかのどのスイッチともインターロックを取りませんので、受信と独立に動作させられます。

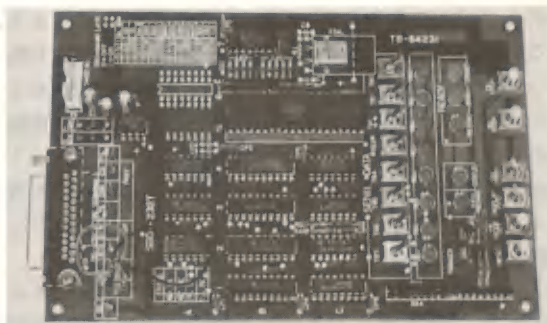
製作

図7に全回路図、図8にプリント・パターンを示します。

製作上難しいところはないと思います。DIP SW_{1,2}を実装するとき、回路No.の順序に気をつけてください。逆に取り付けると、ボーレート表やモード表との対応ができなくなります。

また、プッシュ・スイッチのN.O.とN.C.を逆にするとメチャメチャな動作になります。

IM4712がなくてIM4702を使う人はR₁=10MΩ, C₆=C₇=56pFを忘れずに取り付けてください。

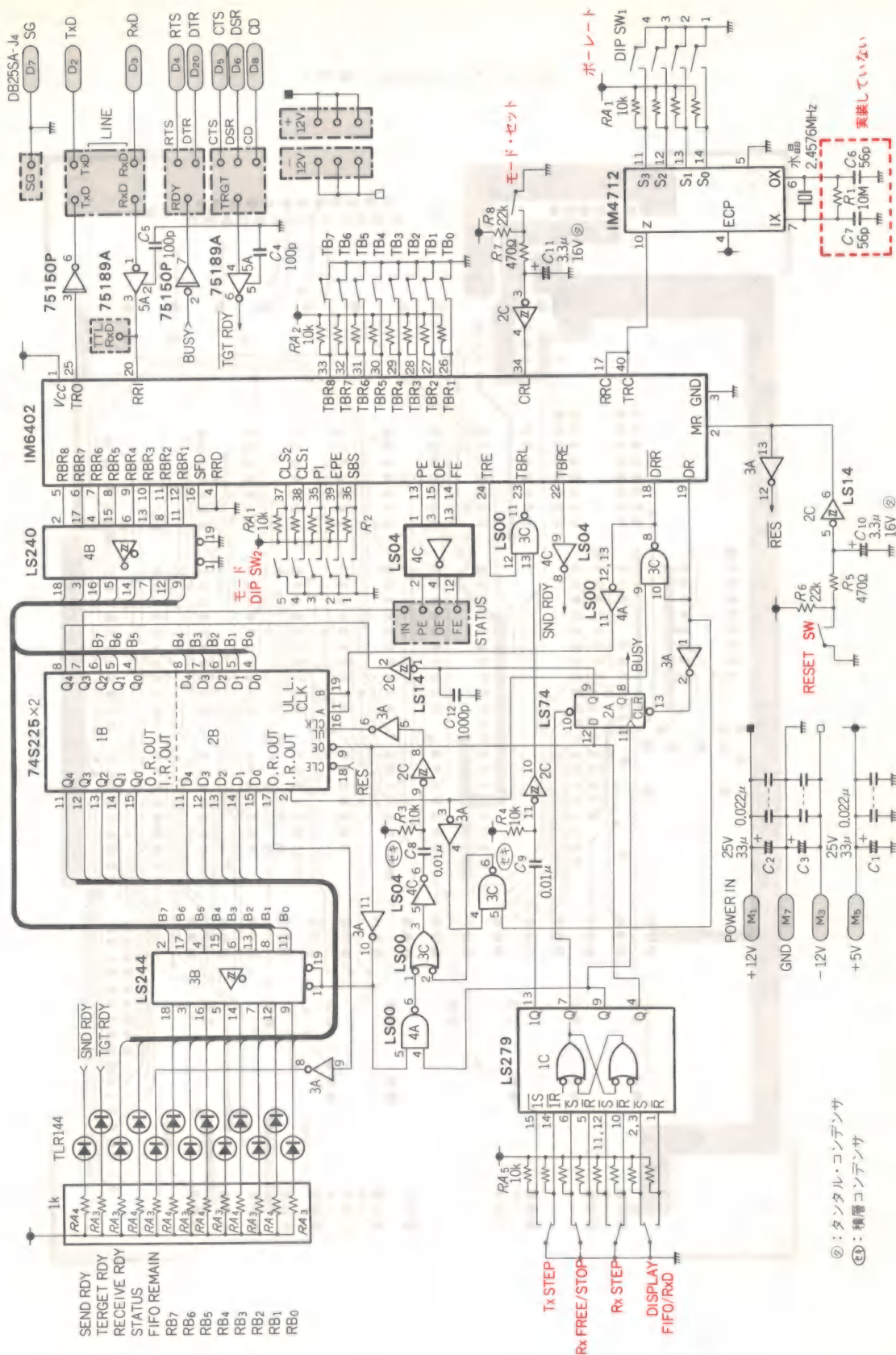


〈写真1〉チェッカの外観

図参考・引用*文献

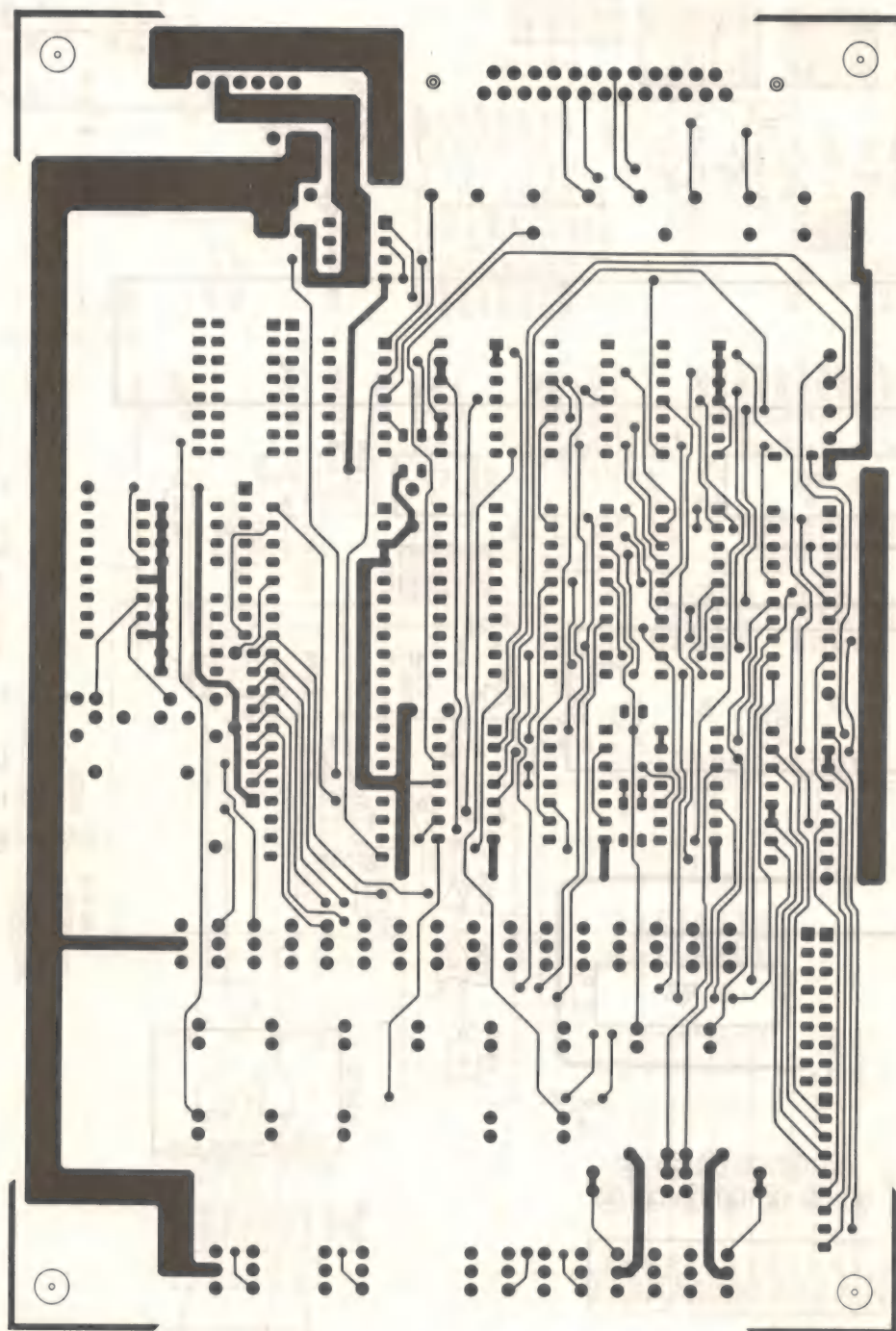
- (1)*神崎康宏：特集*マイコン設計技術の完全マスタ，トランジスタ技術，1985年5月号。
- (2) 森野ひとみ：マイコンとデータ伝送，トランジスタ技術，1983年12月号。
- (3) 特集*マイコン周辺LSI完璧マスタ，トランジスタ技術，1985年3月号。
- (4) 特集*実験で学ぶディジタルIC回路，トランジスタ技術，1984年2月号。
- (5) 石井裕次：シリアル・インターフェースの設計法，トランジスタ技術，1985年11月号。
- (6) 相良富美：周辺制御回路(シリアル・インターフェース)の設計と検討，トランジスタ技術，1982年3月号。
- (7)*PERIPHERAL DESIGN HANDBOOK, INTEL.
- (8)*Z80 SIOテクニカル・マニュアル，シャープ。
- (9)*日本電気，PC9801ユーザーズマニュアル(VF/VM/VX)。
- (10)*日立製作所，日立8ビット・16ビットマイクロコンピュータ周辺LSI。
- (11) Z80周辺LSI活用ノート，インターフェース別冊付録，1983年5月号，p.25。
- (12) 相沢一石：BSCの伝送，トランジスタ技術，1983年12月号，p. 283。
- (13) 安藤善廣：BSC制御手順の詳細と実例，インターフェース，1980年11月号，p. 100，CQ出版社。
- (14) 宮崎誠一：マイクロコンピュータ・データ伝送の基礎と実例，CQ出版社。
- (15)*TI, The Bipolar Digital Integrated Circuits Data Book, 1986。

＜図7＞RS-232Cのチェッカの全回路

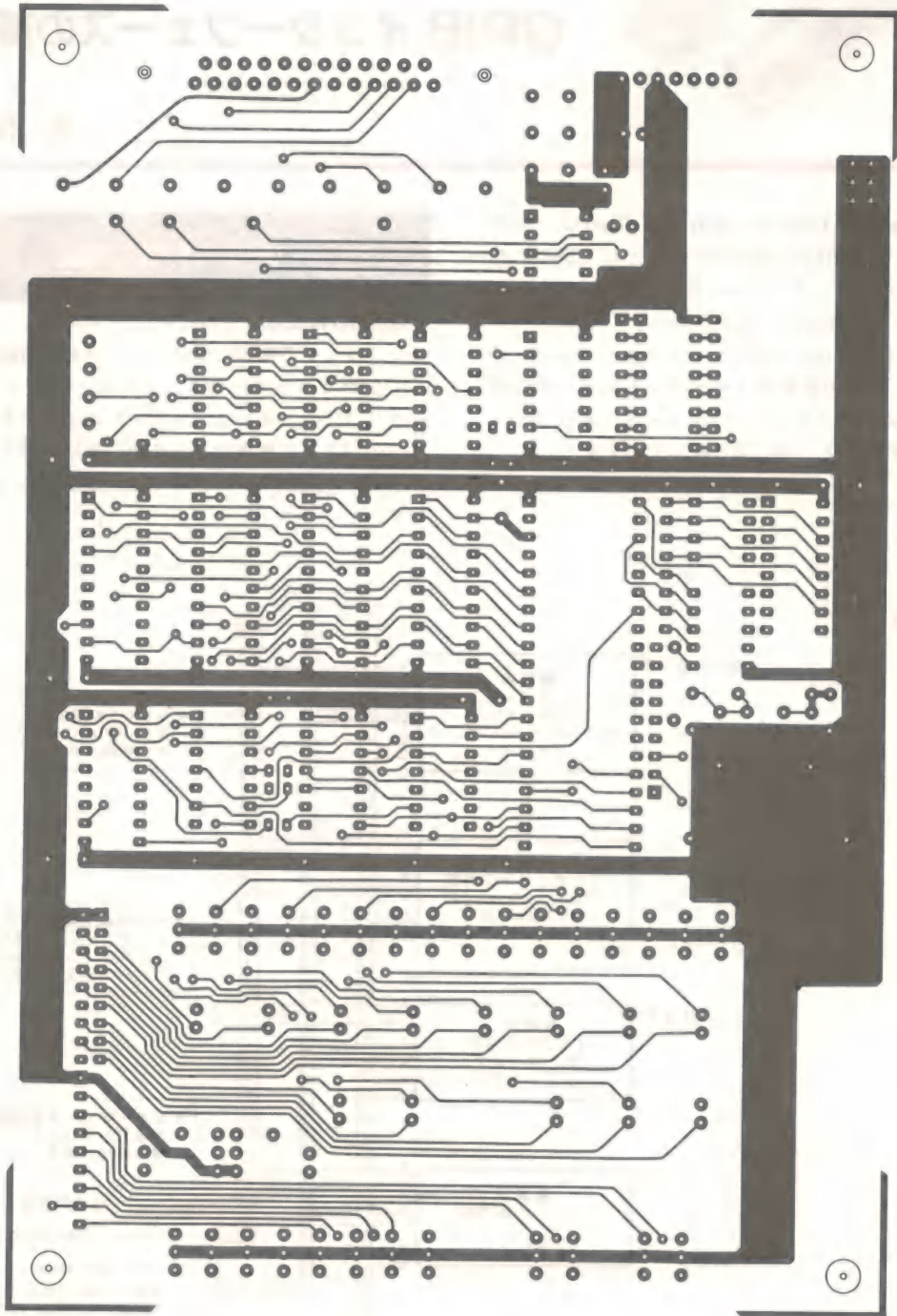


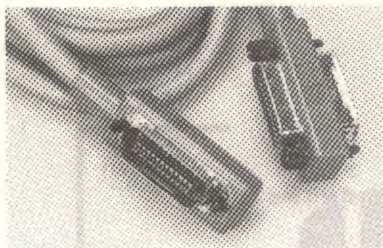
④：タンタル・コンデンサ
⑤：積層コンデンサ

〈図8(a)〉 プリント板のはんだ面(原寸)



〈図 8 (b)〉 プリント板の部品面 (原寸)





§ 3-1

GPIB インターフェースの基礎

里 和政

GPIB(General Purpose Interface Bus)は、米国電気電子学会(IEEE)で1978年に規格として定められたデジタル・インターフェースで、その規格名からIEEE-488とも呼ばれています。

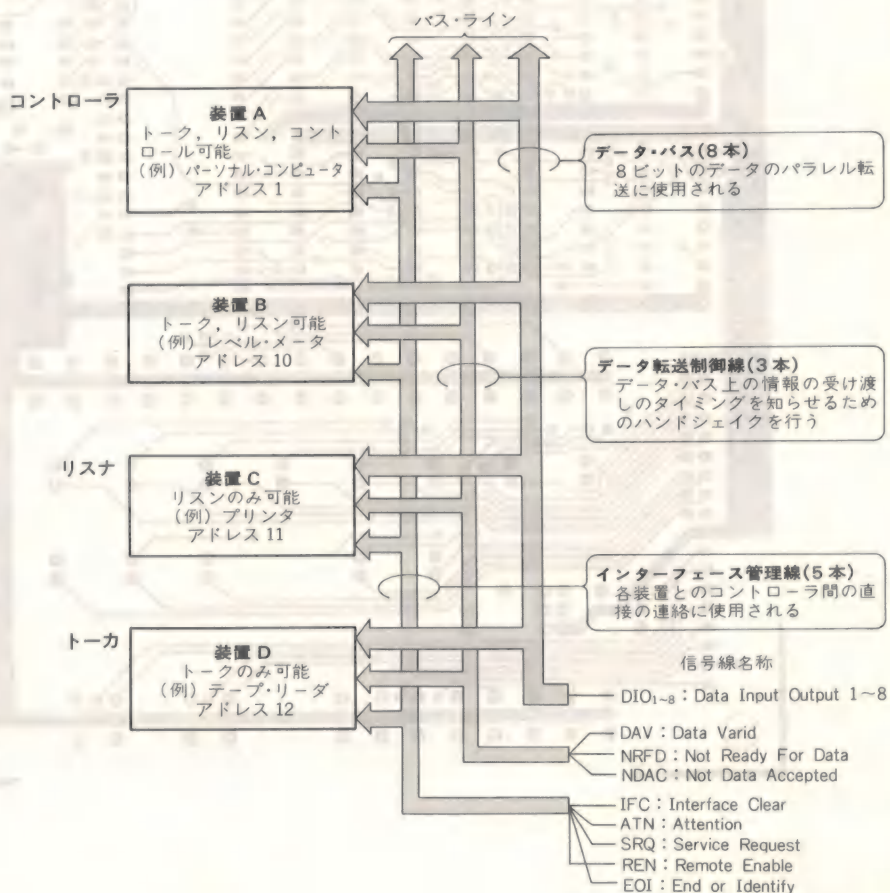
基本となる規格は、米国ヒューレット・パッカード社(HP)が、自社の計測器とパソコンとのデータの通信を行うために考え出したパラレル・インターフェースです。HP社では、HP-IBと呼んでいます。

GPIBの特徴と構成

GPIBの特徴は、そのインターフェース・バス上にハードウェアの増設なしに、いくつもの装置を追加することができることです。それは、CPUのバス上にメモリ、I/Oなどをいくつも接続したような形です。

このバス上に接続できる装置の数は、最大15台までですが、中規模のシステムでは、1回線のバスで十分です。

〈図1〉⁽³⁾
GPIBの構成例



〈表1〉^{(2),(26)}
 GPIBの信号線と
 その機能

信号線		機能	
データ・バス	DIO ₁ (Data Input/Output 1) DIO ₂ (Data Input/Output 2) DIO ₃ (Data Input/Output 3) DIO ₄ (Data Input/Output 4) DIO ₅ (Data Input/Output 5) DIO ₆ (Data Input/Output 6) DIO ₇ (Data Input/Output 7) DIO ₈ (Data Input/Output 8)	データの伝達 (データ例) コマンド アドレス 測定データ ステータス	
	DAV (Data Valid) NRFD (Not Ready For Data) NDAC (Not Data Accepted)	データ有効 受信準備完了 (NRFD=0の時) 受信完了 (NDAC=0の時)	ハンドシェイクを行う
管理線	ATN (Attention) IFC (Interface Clear) SRQ (Service Request) REN (Remote Enable) EOI (End or Identify)	データの区別をする (1: インターフェース・メッセージ) 0: デバイス・メッセージ インターフェースを初期化する サービス要求 リモート/ローカル切り替え データの最終バイトを示す (ATN=0の時) パラレル・ポールの実行を示す (ATN=1の時)	

(注 バスはすべて負論理。0="H", 1="L"レベル)

それらに加えてGPIBは、制御コマンドも決められています。

図1にGPIBの構成を示します。ここで、トーカは送信のみを行う機器であり、バス上に複数のトーカを接続することが可能ですが、同時に複数の動作はできません。

リスナは、受信のみを行う機器であり、これもトーカ同様に複数のリスナをバスに接続することができます。ただし、トーカとは異なり、同時に複数のリスナを動作させることが可能です。

コントローラは、バスの制御を行い、各装置(トーカ, リスナ)からの要求またはデータの送受信を行います。

また、これらの機能をすべてもった装置などもあります。

■ GPIBのバス構成

GPIBバスは、8本のデータ・バス、3本の伝送制御線と5本の管理線の16本から構成されています。表1に各信号線を示します。

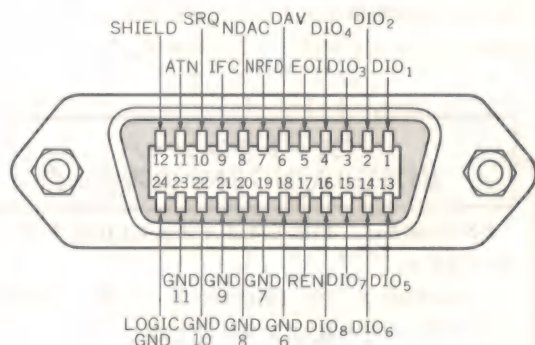
データ・バスは、**双方向性でパラレル**にデータを伝送します。伝送制御線(ハンドシェイク・バス)は、データ・バス上のデータの伝送タイミングおよびデータの方向を制御し、管理線はコントローラが制御する信号で、インターフェースの初期化、データの区別、割り込み、メッセージの管理などを行います。

■ GPIBの電気的特性

GPIBでは、コネクタの形状、信号の制御方法、電気的特性などが規格化されています。図2にコネクタを示します。

電気的特性は、接続できる装置の数、ケーブルの長

〈図2〉^{(2),(26)} GPIBコネクタ



〔例……IEEE規格コネクタ〕
 IEC規格とは異なる

さ、転送速度、装置のバス・ドライブ能力などがあります。

- (1) 装置の接続できる数は、前にも述べたように最大15台です。
- (2) 接続するケーブルの長さは、各装置間で4m以内で、転送データの信頼性などにより、装置の数が11台以上の場合はその総合計が20m以内で、装置が10台以下の場合では、ケーブルの長さは装置の数を2倍した値以下です。
- (3) 転送速度は、1Mバイト/秒以内ですが、装置の接続数によって負荷が異なるため、転送速度は遅くなります。
- (4) GPIBのバス・ドライブには、オープン・コレクタまたは3ステート・ドライブが使用(NRFD, NDAC, SRQはオープン・コレクタのみ)され、レシーバの入力電圧は“L”レベルで0.8V以下、“H”レベルで2.0V以上となっています。

ドライブの出力電圧は、“L”レベルで0.5V以下(電

流シンク +48mA)，“H”レベルで2.4V以上(−5.2mA)となっています。

これらの規格にあった専用のドライバ/レシーバIC (SN75160Aなど)があり、ノイズ・マージンなどもとれ、これらを用いるのがよいでしょう。

GPIOBのケーブルを長く延ばす場合は、コンバータでRS-232Cや光ファイバなどに変換して転送し、再度GPIOBにもどす方法があります。

ハンドシェイクの方法

GPIOBでは、複数の装置を制御するため少し複雑な転送制御を行います。その手順をハンドシェイクと呼び、DAV, NRFD, NDACの3本の制御信号線によってコントロールします。

以下、ハンドシェイクの手順を示します。

DAV：送信されたデータが有効であることを示す。

NRFD：受信準備が完了したことを示す。

NDAC：データを受信したことを示す。

DAVを“H”にし、それによって受信側は、NRFD, NDACを“L”にします。

送信側は、データをデータ・バス上に出力します。受信側すべての装置を受信完了でNRFDが“H”となります。

送信側は、NRFDが“H”となると、DAVを“L”にしてデータを有効にします。受信側は、DAVが“L”になったのちNRFDを“L”にして、データを受信します。受信が完了した時点で、NDACを“H”にします。

送信側は、NDACが“H”になったのちにDAVを“H”にして、次のデータの送信を行います。DAVが“H”になったのち受信側は、NDACを“L”にします。これらによって1バイトの転送が完了します。

この手順を繰り返すことによって、連続転送を行います。図3に転送タイミングを示します。

GPIOBのコマンド

GPIOBは、コントローラからトーカー、リスナにコマ

バス・トランシーバ SN75160 A/MC 3448 のスペック

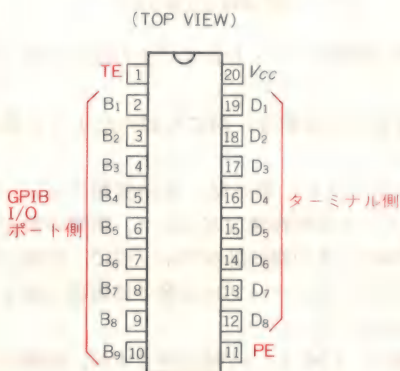
SN75160Aは、TI社の8チャンネルGPIOB用トランシーバです。

バスの制御は、PE, TEピンによって行い、TEピンが“H”のときICはドライバとなり、PEピンが

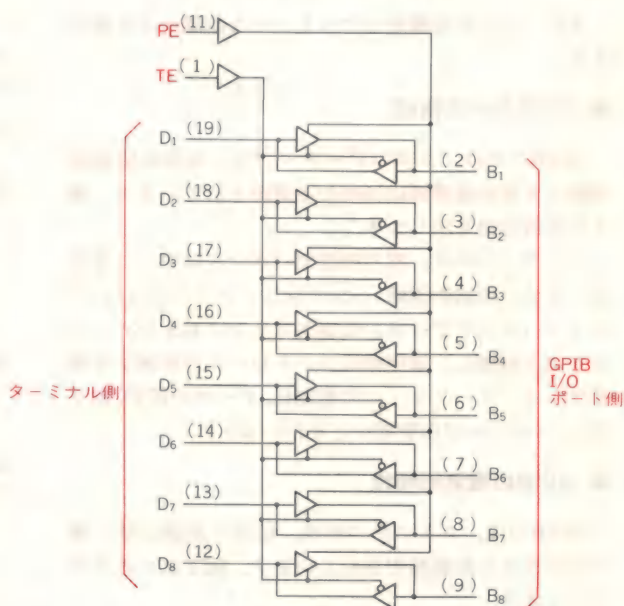
“H”で出力できます。TEピンが“L”のときは、レシーバとなります。図Aにピン配置、図Bに内部構成を、表Aに電気的特性を示します。

ほかに、同様のICとしてSN75162Aがあります。このICは、SN75160Aとほぼ同一仕様ですが、ピンが

〈図A〉⁽⁷⁾ SN75160Aのピンの配置

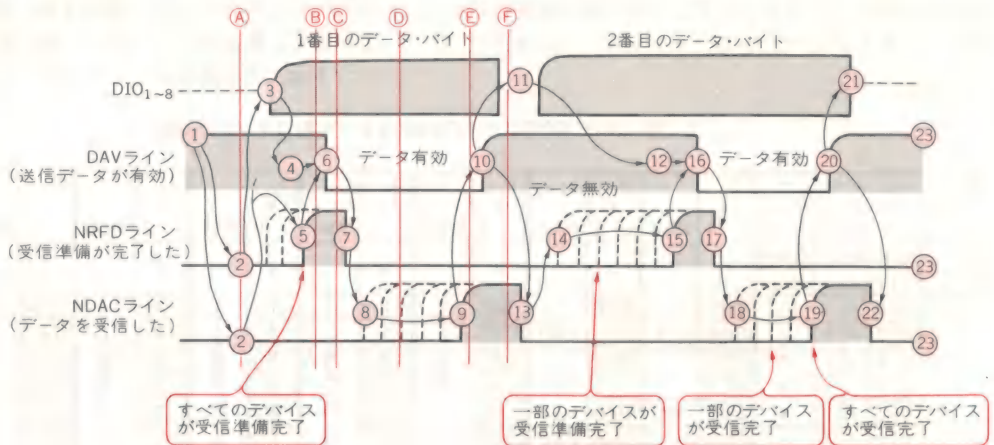


〈図B〉⁽⁷⁾ SN75160Aの内部構造



〈図3〉^{(2),(26)}

GPIB の3線ハ
ンドシェイクの
タイムチャート



- (a) 送信を開始する装置はDAVを“H”にする ①
それにより受信する装置は、NRFD、NDACを“L”にする ②
(b) 送信するデータをバス上に出力する ③、④
すべての装置が受信準備を完了すると、NRFDが“H”になり ⑤ DAVを“L”にしてデータを有効にする ⑥
(c) 受信装置は、DAVが“L”になるとNRFDを“L”にしてデータを受信する ⑦
受信が完了すると、NDACを“H”にする ⑨
(d) 受信完了によりDAVを“H”にして、次のデータを送信する ⑩
DAVが“H”になるとNDACを“L”にして、次のデータ受信をする ⑬
以上の繰り返しによってデータの送受信を行う

GPIBバスと同じ形式となっています(図C参照)。

MC3448は、モトローラ社の4チャンネルGPIB用ト
ランシーバです。

SN75160Aと同じように Send/Rec.,Enableピンに
よって制御します。Send/Rec.ピンは、“H”のときドラ
イバに、“L”のときレシーバになります。Enableピンは、
ドライバ動作時に“L”で出力されます。

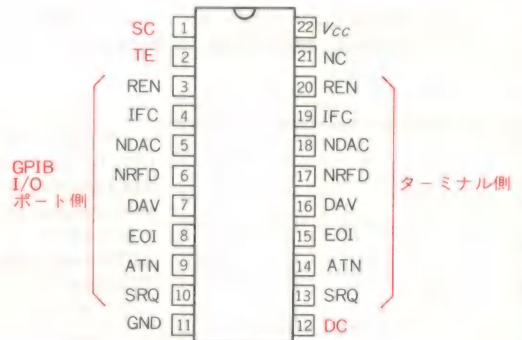
図Dにピン配置を示します。

〈表A〉
SN75160Aの
電気的特性

電源電圧	+5V
消費電力	66mW
ドライバ部	オープン・コレクタ
ドライバ電流	48mA
レシーバ・ヒステリシス	650mV

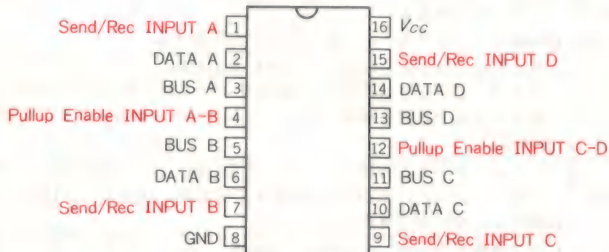
〈図C〉⁽⁷⁾ SN75162Aのピンの配置

(TOP VIEW)



〈図D〉⁽⁶⁾ MC3448Aのピンの配置

(TOP VIEW)



電源電圧	+5V
ドライバ・ドライブ電流	48mA
レシーバ・ヒステリシス	400mV

コマンドは7ビットで構成され、128種類から成っています。表2(a)にコマンド一覧を示します。またコマンドは、その機能によってグループごとに区別され

<div> b_7 b_6 b_5 </div>					<div> 0 0 </div>		<div> 0 0 </div>		<div> 0 1 </div>		<div> 0 1 </div>		<div> 1 0 </div>		<div> 1 0 </div>		<div> 1 1 </div>		<div> 1 1 </div>	
					MSG		MSG		MSG		MSG		MSG		MSG		MSG			
ビット					Column Row↓															
0	0	0	0	0	NUL		DCE		SP	↑	0	↑	@	↑	P	↑	.	↑	p	↑
0	0	0	1	1	SOH	GTL	DC1	LLO	!	1	1	A	↑	Q	↑	a	↑	q	↑	
0	0	1	0	2	STX		DC2		#	2	2	B	↑	R	↑	b	↑	r	↑	
0	0	1	1	3	ETX		DC3		#	3	3	C	↑	S	↑	c	↑	s	↑	
0	1	0	0	4	EOT	SDC	DC4	DCL	\$	4	4	D	↑	T	↑	d	↑	t	↑	
0	1	0	1	5	ENQ	PPC	NAK	PPU	%	5	5	E	↑	U	↑	e	↑	u	↑	
0	1	1	0	6	ACK		SYN		&	6	6	F	↑	V	↑	f	↑	v	↑	
0	1	1	1	7	BEL		ETB		'	7	7	G	↑	W	↑	g	↑	w	↑	
1	0	0	0	8	BS	GET	CAN	SPE	(8	8	H	↑	X	↑	h	↑	x	↑	
1	0	0	1	9	HT	TCT	EM	SPD)	9	9	I	↑	Y	↑	i	↑	y	↑	
1	0	1	0	10	LF		SUB		*	10	10	J	↑	Z	↑	j	↑	z	↑	
1	0	1	1	11	VT		ESC		+	11	11	K	↑	\	↑	k	↑		↑	
1	1	0	0	12	FF		FS		,	12	12	L	↑]	↑	l	↑		↑	
1	1	0	1	13	CR		GS		-	13	13	M	↑	^	↑	m	↑		↑	
1	1	1	0	14	SO		RS		.	14	14	N	↑	~	↑	n	↑	~	↑	
1	1	1	1	15	SI		US		/	15	15	O	↑	_	↑	o	↑	DEL	↑	

アドレス・ユニバーサル・グループ

コマンド・グループ

リスナ・アドレス・グループ

グループ

トーカ・アドレス・グループ

グループ

二次コマンド・グループ

(SCG)

一次コマンド・グループ (PCG)

(注2) $b_1 \sim b_7$ は, $DIO_1 \sim DIO_7$ に順番に対応する.

① アドレス・コマンド・グループ	<p>アドレス・コマンドは、リスナまたはトーカに指定されている装置が機能するコマンドです。</p> <p>GTL(Go To Local)：ローカル状態になる</p> <p>SDC(Selected Device Clear)：装置の初期化を行う</p> <p>PPC(Parallel Poll Configure)：二次コマンドと併用し、パラレル・ポールのライン割り振りを行う</p> <p>GET(Group Execute Trigger)：トリガ(測定開始)を受けることになる</p> <p>TCT(Take Control)：トーカに指定され、このコマンドを受信すると、その装置が以後コントローラになる</p>
② ユニバーサル・コマンド・グループ	<p>ユニバーサル・コマンドは、バスに接続されているすべての装置に対して送られるコマンドです。</p> <p>LLO(Local Lockout)：ローカル・ロック状態になり、装置側からリモート/ローカル状態を変えられなくなる</p> <p>DCL(Device Clear)：すべての装置が初期化を行う</p> <p>PPU(Parallel Poll Unconfigure)：すべての装置のパラレル・ポールの設定をクリアする</p> <p>SPE(Serial Poll Enable)：GPIOをシリアル・ポール・モードにする。このモード中は、データの代わりにステータスを送る</p> <p>SPD(Serial Poll Disable)：シリアル・ポール・モードを解除する</p>
③ リスナ・アドレス・グループ	<p>リスナ・アドレスは、コントローラがある装置をリスナにするため、コマンドにアドレス番号を付けて送ります。リスナ・アドレスは20Hから3FHまでですが、3FHだけはUNLコマンドとして、すべての装置のリスナ状態を解除します。したがって、装置のアドレス番号は00Hから1EHまでの31種類です。</p>
④ トーカ・アドレス・グループ	<p>トーカ・アドレスはコントローラがある装置をトーカにするため、コマンドにアドレス番号を付けて送ります。トーカ・アドレスは40Hから5FHまでですが、5FHだけはUNTコマンドとして、すべての装置のトーカ状態を解除します。</p>
⑤ 二次コマンド・グループ	<p>二次コマンドは、これまで説明したコマンドに伴って送られ、トーカ/リスナ・アドレスを拡張したり、パラレル・ポールのライン割り振りに使用します。</p>

ています(コマンド説明参照)。動作の対象となる装置は、コマンドによって1台または複数台となります。表2(b)にコマンドの送信シーケンスを示します。

コマンドを送る場合は、ATN信号を“H”にして装置にコマンドであることを認識させます。データの場合は、ATN信号は、“L”です。

〈表2(b)〉⁽³⁾

コマンド送信のシーケンス

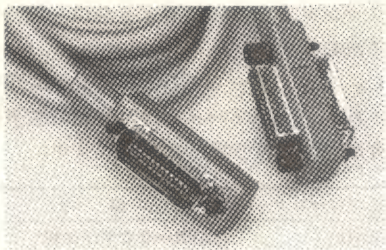
- (注) ・LADは リスナ・アドレス、必要により2次アドレスが続く
 ・TADは トーカ・アドレス、必要により2次アドレスが続く
 ・UNLは アンリスナ
 ・UNTは アントーカ
 ・CLAは コントローラのリスナ・アドレス

項 目	ATN=1のコマンド	備 考
データ転送	UNL, TAD, LAD	ATN=0でデータ列
リモート	UNL, LAD	事前にREN=1にする
ローカル	UNL, LAD, GTL	事前にREN=1にする
ローカル・ロックアウト	UNL, LLO	事前にREN=1にする
トリガ	UNL, LAD, GET	
デバイス・クリア	UNL, DCL	全装置が対象
デバイス・クリア	UNL, LAD, SDC	指定装置が対象
コントローラ委譲	TAD, TCT	
シリアル・ボール実行	UNL, SPE, CLA, TAD	ATN=0でステータス・バイト
シリアル・ボール終了	UNL, SPD	
パラレル・ボールの設定	UNL, LAD, PPC, PPE, UNL	
パラレル・ボールの解除	PPU	全装置が対象
パラレル・ボールの解除	UNL, LAD, PPC, PPE, UNL	指定装置が対象
パラレル・ボールの実行	パラレル・ボールの応答データ	ATN=1に加えてEOI=1

- | | |
|------------------------------|----------------------------------|
| ・GTL (Go To Local) | ・PPD (Parallel Poll Disable) |
| ・LLO (Local Lockout) | ・SPE (Serial Poll Enable) |
| ・GET (Group Execute Trigger) | ・SPD (Serial Poll Disable) |
| ・DCL (Device Clear) | ・PPC (Parallel Poll Configure) |
| ・SDC (Selected Device Clear) | ・PPE (Parallel Poll Enable) |
| ・TCT (Take Control) | ・PPU (Parallel Poll Unconfigure) |

〈表3〉^{(2),(26)} GPIBのファンクション

SHファンクション	DIOライン上のメッセージを確実に送信する機能。
AHファンクション	DIOライン上のメッセージを受信する機能。 バスに接続されたSHファンクションと、AHファンクションとのハンドシェイク・シーケンスによって、メッセージの非同期転送が行われる。
TまたはTEファンクション	装置がトーカとしてアドレス指定されたとき、その他の装置へデータをインターフェースを通じて送る機能。TEファンクションは、2次アドレスにより拡張トーカとして機能する。
LまたはLEファンクション	装置がリスナとしてアドレス指定されたとき、その他の装置からデータをインターフェースを通して受け取る機能。LEファンクションは、2次アドレスにより拡張リスナとして機能する。
SRファンクション	インターフェースを管理するコントローラへ、非同期のサービスを要求する機能。コントローラのシリアル・ポーリングにより、装置はステータス・バイトを送出する。
RLファンクション	装置が、リモート動作かローカル動作かを選択する機能。
PPファンクション	装置が、コントローラのパラレル・ボール時に、トーカに指定されることなく、コントローラに対して1ビットのステータスを送出する機能。DIOライン1本に1装置を割り当てることで、一度に8台の装置まで可能。 コントローラへのサービス要求は、SRQメッセージを用いたものと、パラレル・ボールによるものがあります。前者は、装置がサービス要求をしたい場合、非同期でコントローラに送信可能ですが、1本の信号ラインでOR接続されているため、どの装置から要求があったのか判断できないといった短所があります。後者は、サービス要求をしている装置を、同時に8台まで認識できますが、コントローラが必要と感じたときのみで、装置がサービス要求したいときには、サービス要求ができないといった欠点があります。システムにより使い分けが必要ですが、一般的にSRQメッセージを使用したものが多く見られます。
DCファンクション	装置の初期化をする機能。
DTファンクション	リスナに指定された装置を、測定開始(または動作開始)する機能。
Cファンクション	インターフェースを通じて、装置のアドレス、コマンド、その他の装置に送る機能。またどのデバイスがサービスを要求しているかのシリアル・ボール、パラレル・ボールを行う機能を持つコントローラとしての機能 などです



§ 3-2

GPIB コントロール LSI の 使い方

里 和政/松井雅行/竹尾佳己

GPIBを制御するためのハードウェアは、複雑なうえソフトウェアの負担を考え合わせると、専用のLSIを使用するほうが簡単になります。

各メーカーから専用のLSIが発表されており、その中でも代表的なLSIについて説明します。

μPD7210

μPD7210は、IEEE-488の規格のインターフェース機能をもっています。PC9801のGPIB用のコントローラでも使用されています。

● μPD7210の特性

このLSIは、トーカー、リスナ、コントローラの機能をプログラムによって制御することができます。また、データ転送に必要なハンドシェイクも自動的に行えます。これらの制御には、16個のライト/リード・レジスタを使用します。

図1にピン配置を、図2に内部ブロックを、図3にコントロール・レジスタを示します。図4に電気的特性およびタイムチャートを示します。

● 応用回路例

図5に、μPD7210を80系バスに接続するときの回路図を示します。この図のI/Oアドレスは、80H～87H番地までをコントローラに、88Hまたは89H番地をアドレス・スイッチ(S₁)データ・リード用に使用しています。また、DMAREQ端子を割り込み(DI/DO割り込み)に使用しています。

S₂スイッチを+5V側にするにより、システム・コントローラとして使用します。

■ μPD7210のソフトウェア

GPIB用LSIを使用するためのソフトウェアは、電源投入時の初期設定、LSIから装置への割り込み要求、装置からLSIへの処理の三つに分類できます。

ここではμPD7210を使用してトーカー、リスナ機能を実現する場合を例にして説明します⁽¹²⁾。レジスタについては図3を参照してください。

● 初期設定

μPD7210は、電源投入時に補助コマンドのponメッセージによりpon状態にし、初期設定を行ってからpon状態を解除することにより、初めて使用可能となります。

初期設定ではつぎのことを行います。

- ① 割り込みマスク・レジスタの設定
- ② シリアル・ボール・モードのクリア
- ③ アドレス・モード・レジスタの設定
- ④ アドレス・レジスタの設定
- ⑤ クロック周波数の設定
- ⑥ データ受信モードの設定
- ⑦ T₁ディレイ時間の設定

T₁ディレイ時間は、通常2μs以上です。ただし、DIO_{1~8}、DAV、EOIラインに3ステート・ドライバを使用している場合には1100ns以上で、かつ2バイト目以降は500ns以上でよいことになっています。

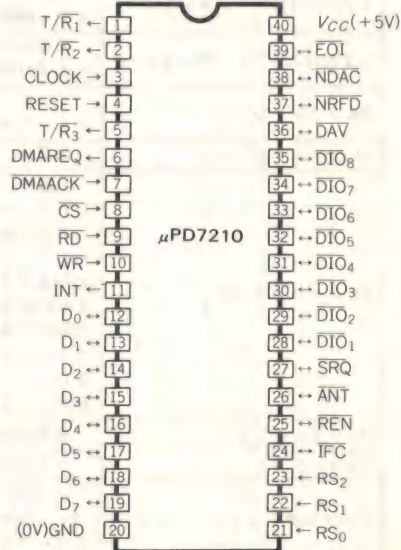
- ⑧ DC、DTファンクションのホールド・オフの設定

初期設定のフローチャートを図6に示します。

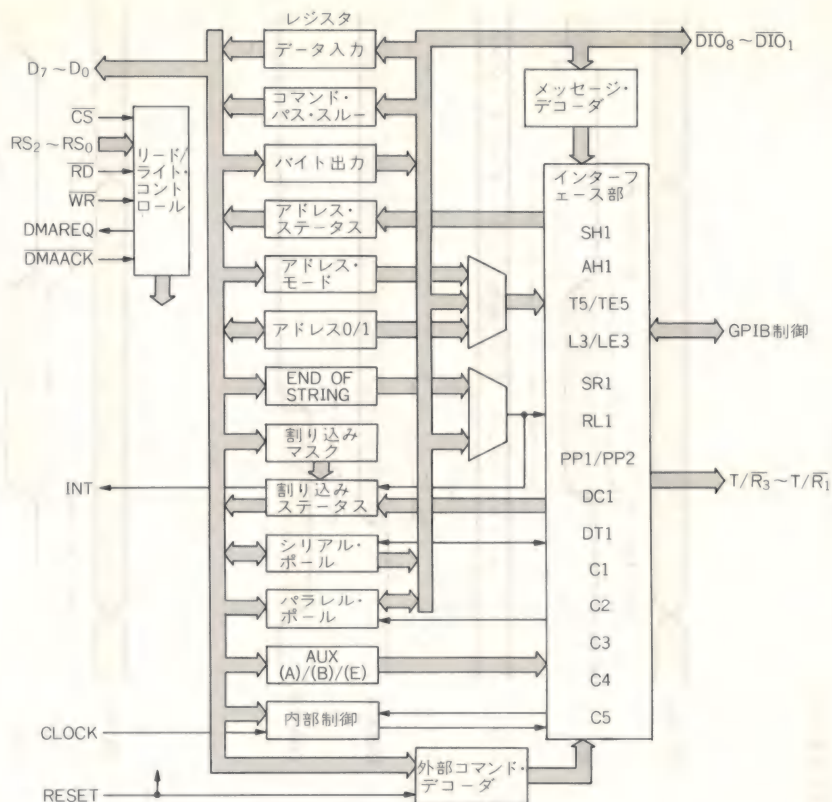
● 割り込み処理

μPD7210から装置への割り込み要求は13種類ありますが、初期設定ではつぎの7種類の割り込みをイネ

〈図1〉⁽⁴⁾
μPD7210の
ピン配置



〈図2〉⁽⁴⁾
 μ PD7210の内部ブロック



〈図3〉⁽⁴⁾ μ PD7210の内部レジスタ

リード・レジスタ

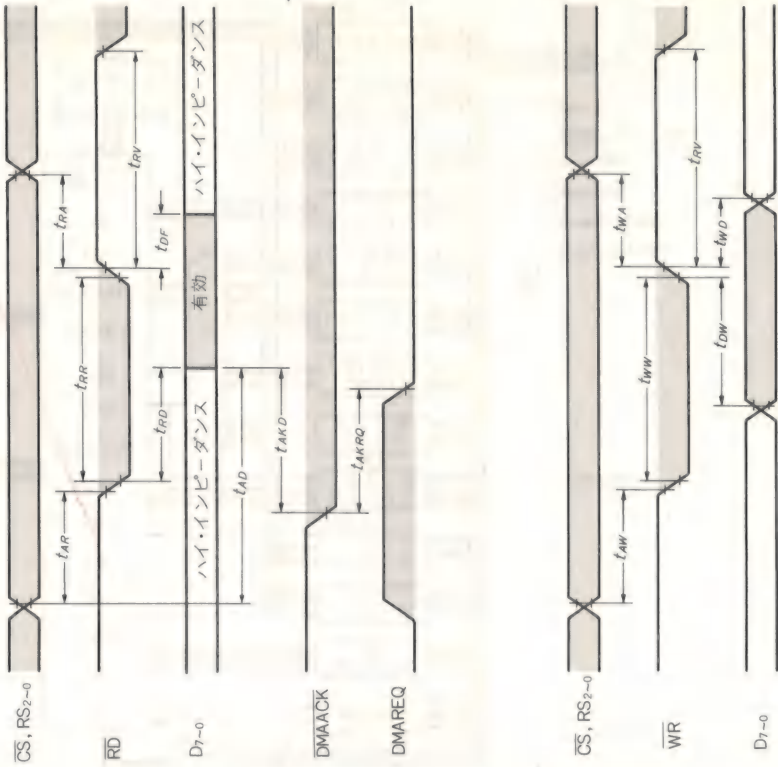
RS ₂ RS ₁ RS ₀	レジスタ	説明
(0R)	DI ₇ DI ₆ DI ₅ DI ₄ DI ₃ DI ₂ DI ₁ DI ₀	データ入力
(1R)	CPT APT DET END DEC ERR DO DI	割り込みステータス 1
(2R)	INT SRQI LOK REM CO LOKC REMC ADSC	割り込みステータス 2
(3R)	S ₈ PEND S ₆ S ₅ S ₄ S ₃ S ₂ S ₁	シリアル・ボール・ステータス
(4R)	CIC ATN SPMS LPAS TPAS LA TA MJMH	アドレス・ステータス
(5R)	CPT ₇ CPT ₆ CPT ₅ CPT ₄ CPT ₃ CPT ₂ CPT ₁ CPT ₀	コマンド・バス・スルー
(6R)	× DT ₀ DL ₀ AD ₅₋₀ AD ₄₋₀ AD ₃₋₀ AD ₂₋₀ AD ₁₋₀	アドレス 0
(7R)	EOI DT ₁ DL ₁ AD ₅₋₁ AD ₄₋₁ AD ₃₋₁ AD ₂₋₁ AD ₁₋₁	アドレス 1

ライト・レジスタ

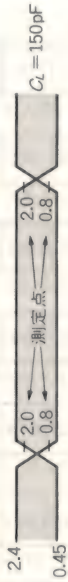
RS ₂ RS ₁ RS ₀	レジスタ	説明
(0W)	BO ₇ BO ₆ BO ₅ BO ₄ BO ₃ BO ₂ BO ₁ BO ₀	バイト出力
(1W)	CPT APT DET END DEC ERR DO DI	割り込みマスク 1
(2W)	0 SRQI DMAO DMAI CO LOKC REMC ADSC	割り込みマスク 2
(3W)	S ₈ rsv S ₆ S ₅ S ₄ S ₃ S ₂ S ₁	シリアル・ボール・モード
(4W)	ton lon TRM ₁ TRM ₀ 0 0 ADM ₁ ADM ₀	アドレス・モード
(5W)	CNT ₂ CNT ₁ CNT ₀ COM ₄ COM ₃ COM ₂ COM ₁ COM ₀	AUX モード
(6W)	ARS DT DL AD ₅ AD ₄ AD ₃ AD ₂ AD ₁	アドレス 0/1
(7W)	EC ₇ EC ₆ EC ₅ EC ₄ EC ₃ EC ₂ EC ₁ EC ₀	End of String

〈図 4 (a)〉⁽⁴⁾ μ P7210のタイムチャート

項 目	記 号	条 件	min	max
$\overline{\text{EOI}} \downarrow \rightarrow \text{DIO}$ 遅延時間	t_{EODI}	PPSS \rightarrow PPAS, ATN = True		250
$\overline{\text{EOI}} \downarrow \rightarrow \text{T/R}_1$ 遅延時間	t_{EOTI1}	PPSS \rightarrow PPAS, ATN = True		155
$\overline{\text{EOI}} \uparrow \rightarrow \text{T/R}_1$ 遅延時間	t_{EOTI2}	PPAS \rightarrow PPSS, ATN = False		200
$\text{ATN} \downarrow \rightarrow \text{NDAC}$ 遅延時間	t_{ATND}	AIDS \rightarrow ANRS, LIDS		155
$\text{ATN} \downarrow \rightarrow \text{T/R}_1$ 遅延時間	t_{ATT1}	TACS + SPAS \rightarrow TADS, CIDS		155
$\text{ATN} \downarrow \rightarrow \text{T/R}_2$ 遅延時間	t_{ATT2}	TACS + SPAS \rightarrow TADS CIDS		200
$\text{DAV} \downarrow \rightarrow \text{DMAREQ}$ 遅延時間	t_{DVRQ}	ACRS \rightarrow ACDS LACS		600
$\text{DAV} \downarrow \rightarrow \text{NRFD}$ 遅延時間	t_{DVNR1}	ACRS \rightarrow ACDS		350
$\text{DAV} \downarrow \rightarrow \text{NDAC}$ 遅延時間	t_{DVND1}	ACRS \rightarrow ACDS \rightarrow AWNS		650
$\text{DAV} \uparrow \rightarrow \text{NDAC}$ 遅延時間	t_{DVND2}	AWNS \rightarrow ANRS		350
$\text{DAV} \uparrow \rightarrow \text{NRFD}$ 遅延時間	t_{DVNR2}	AWNS \rightarrow ANRS \rightarrow ACRS		350
$\text{RD} \downarrow \rightarrow \text{NRFD}$ 遅延時間	t_{RNR}	ANRS \rightarrow ACRS LACS, DI reg. selected		500
$\text{NDAC} \uparrow \rightarrow \text{DMAREQ}$ 遅延時間	t_{NDRQ}	STRS \rightarrow SWNS \rightarrow SGNS, TACS		400
$\text{NDAC} \uparrow \rightarrow \text{DAV}$ 遅延時間	t_{NDDV}	STRS \rightarrow SWNS \rightarrow SGNS		350
$\text{WR} \uparrow \rightarrow \text{DIO}$ 遅延時間	t_{WDI}	SGNS \rightarrow SDYS, BO reg. selected		250
$\text{NRFD} \uparrow \rightarrow \text{DAV}$ 遅延時間	t_{NRDV}	SDYS \rightarrow STRS, T ₁ = True		350
$\text{WR} \uparrow \rightarrow \text{DAV}$ 遅延時間	t_{WDV}	SGNS \rightarrow SDYS \rightarrow STRS BO reg. selected, RFD = True N _F = f _c = 8 MHz, T ₁ (高速)		$t_{\text{SYNC}} + 830$
TRIG パルス幅	t_{TT}		50	



ACテスト入出力波形



〈図 4 (b)〉^(a) μ PD7210の電気的特性

(a) 絶対最大定格 ($T_a = 25^\circ\text{C}$)

項 目	記 号	定 格	単 位
電源電圧	V_{CC}	$-0.5 \sim 7.0$	V
入力電圧	V_I	$-0.5 \sim 7.0$	V
出力電圧	V_O	$-0.5 \sim 7.0$	V
動作温度	T_{opt}	$0 \sim +70$	$^\circ\text{C}$
保存温度	T_{stg}	$-65 \sim +150$	$^\circ\text{C}$

(b) DC特性 ($T_a = 0^\circ\text{C} \sim +70^\circ\text{C}$, $V_{CC} = +5\text{V} \pm 10\%$)

項 目	記 号	条 件	min	max	単 位
低レベル入力電圧	V_{IL}		-0.5	0.8	V
高レベル入力電圧	V_{IH}		2.0	$V_{CC} + 0.5$	V
低レベル出力電圧	V_{OL}	$I_{OL} = 2\text{mA}$, T/R_1 以外		0.45	V
		$I_{OL} = 4\text{mA}$, T/R_1		0.45	V
高レベル出力電圧	V_{OH1}	$I_{OH} = -400\mu\text{A}$, INT以外	2.4		V
		$I_{OH} = -400\mu\text{A}$ — INT	2.4		V
		$I_{OH} = -50\mu\text{A}$	3.5		V
入力リーク電流	I_{LI}		-10	10	μA
出力リーク電流	I_{LO}		-10	10	μA
電源電流	I_{CC}			180	mA

(c) 容量 ($T_a = 25^\circ\text{C}$, $V_{CC} = \text{GND} = 0\text{V}$)

項 目	記 号	条 件	max	単 位
入力容量	C_{IN}	$f = 1\text{MHz}$ 被測定端子以外は 0V	10	pF
出力容量	C_{OUT}		15	pF
入出力容量	$C_{I/O}$		20	pF

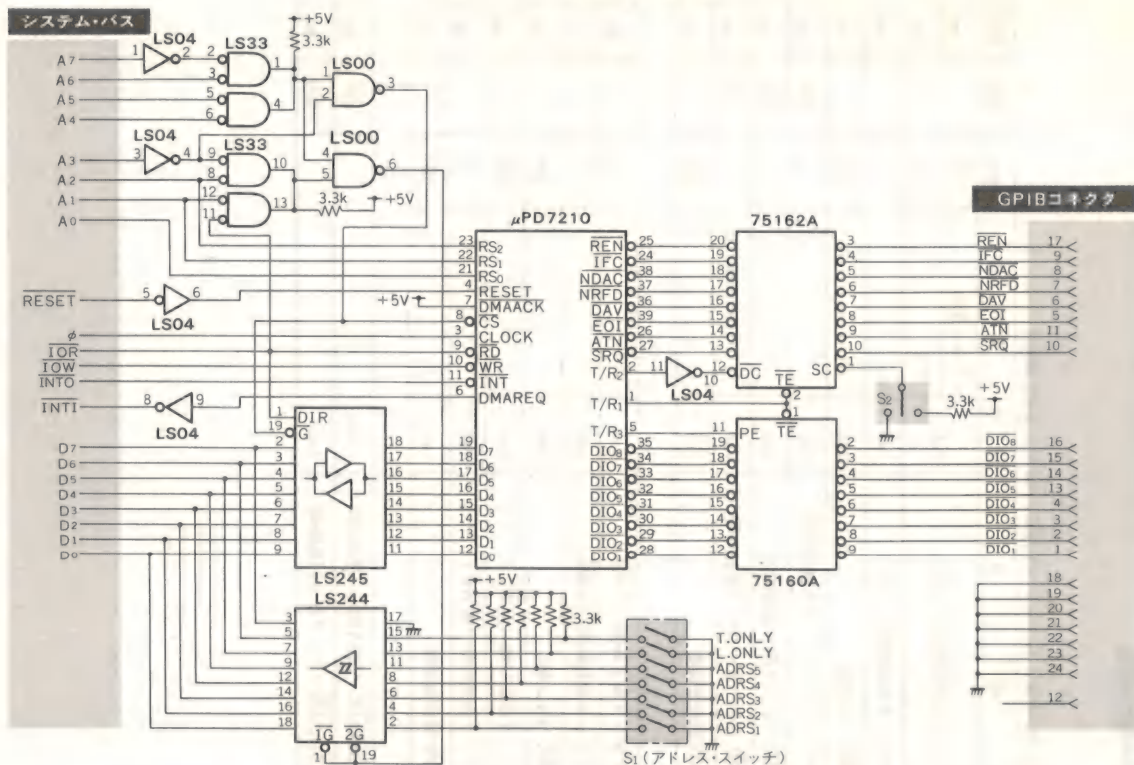
(d) AC特性 ($T_a = 0^\circ\text{C} \sim +70^\circ\text{C}$, $V_{CC} = +5\text{V} \pm 10\%$)

項 目	記 号	条 件	min	max	単 位
アドレス設定時間 (対RD)	t_{AR}	$RS_0 \sim RS_2$	85		ns
アドレス保持時間 (対RD)	t_{RA}	CS	0		ns
\overline{RD} パルス幅	t_{RR}		170		ns
アドレス \rightarrow データ出力遅延時間	t_{AD}			250	ns
RD \rightarrow データ出力遅延時間	t_{RD}			150	ns
RD \rightarrow データ・フロー・ポート遅延時間	t_{DF}		0	80	ns
RD間回復時間	t_{RV}		250		ns

アドレス設定時間 (対WR)	t_{AW}		0		ns
アドレス保持時間 (対WR)	t_{WA}		0		ns
WR パルス幅	t_{WW}		170		ns
データ設定時間 (対WR)	t_{DW}		150		ns
データ保持時間 (対WR)	t_{WD}		0		ns
WR間回復時間	t_{RV}		250		ns

$\overline{\text{DMAACK}} \rightarrow \overline{\text{DMAREQ}}$ 遅延時間	t_{AKRQ}			130	ns
$\overline{\text{DMAACK}} \rightarrow$ データ出力遅延時間	t_{AKD}			200	ns

〈図5〉⁽¹⁾ μ PD7210の応用回路図



ープルにしました。

① DI(Data In)

バイト入力レジスタにデータ・バイトが蓄えられていることを示しています。リスナに指定され、アクセプタ・ハンドシェイクを行おうとしています。

② DO(Data Out)

バイト出力レジスタへのデータ・バイトの書き込み要求、トーカに指定され、ソース・ハンドシェイクを行おうとしています。

③ DEC(Device Clear)

DCLまたはSDCコマンドを受信した。

④ DET(Device Trigger)

GETコマンドを受信した。

⑤ ADSC(Address Status Change)

トーカまたはリスナのアクティブ/アイドル状態が遷移した。

⑥ REMC(Remote Change)

リモート/ローカル状態が遷移した。

⑦ LOKC(Lockout Change)

リモート/ローカルのロック状態が遷移した。

割り込み処理のフローチャートを図7に示します。

● 装置の処理

トーカ、リスナ機能では、装置から μ PD7210への処理にはつぎの処理があります。

① データ受信

リスナに指定されたことにより、データの受信を行う。

② データ送信

トーカに指定されたことにより、データの送信を行う。

③ サービス要求

コントローラに対してサービスを要求する。

④ ローカル状態に設定

装置のローカル・スイッチが押されたことにより、LSIをローカル状態にする。

装置の処理のフローチャートを図8に示します。またプログラム例をリスト1(pp.92~94)に示します。

MC68488

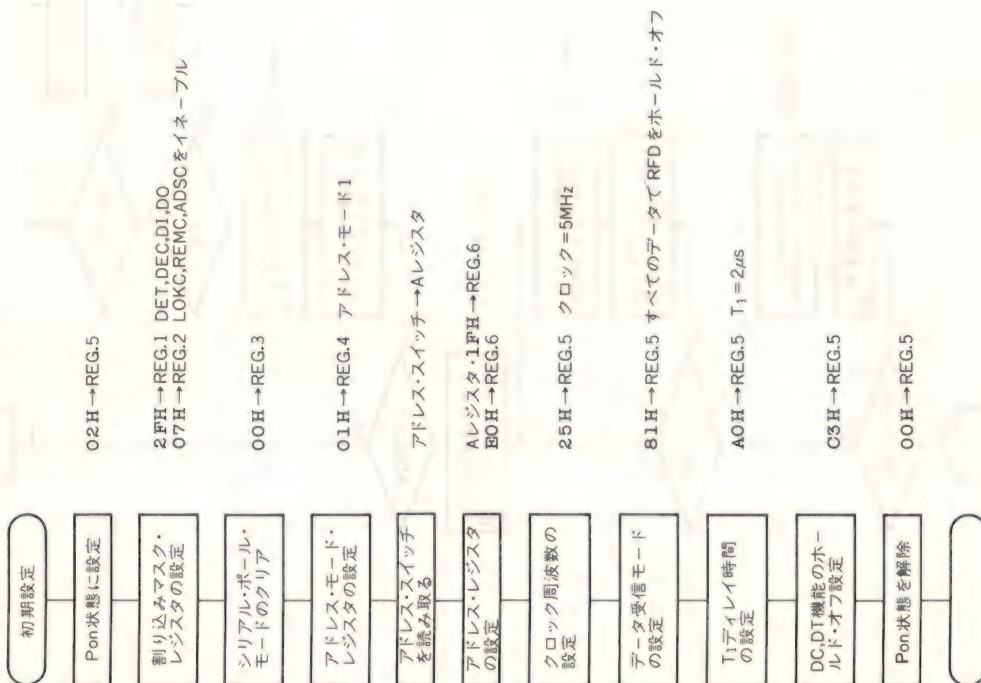
MC68488は、68系のモトローラ社のGPIA(General Purpose Interface Adapter)で、かなり早い時期に発表されました。

● MC68488の特徴

このLSIは、トーカ、リスナの機能だけでコントローラとしての機能は、付加されていません。もっぱら計測器に組み込むためのものです。

そのためコントローラとして使用する場合は、

＜図6＞ 初期設定のフローチャート



＜図7＞ 割り込み処理フローチャート

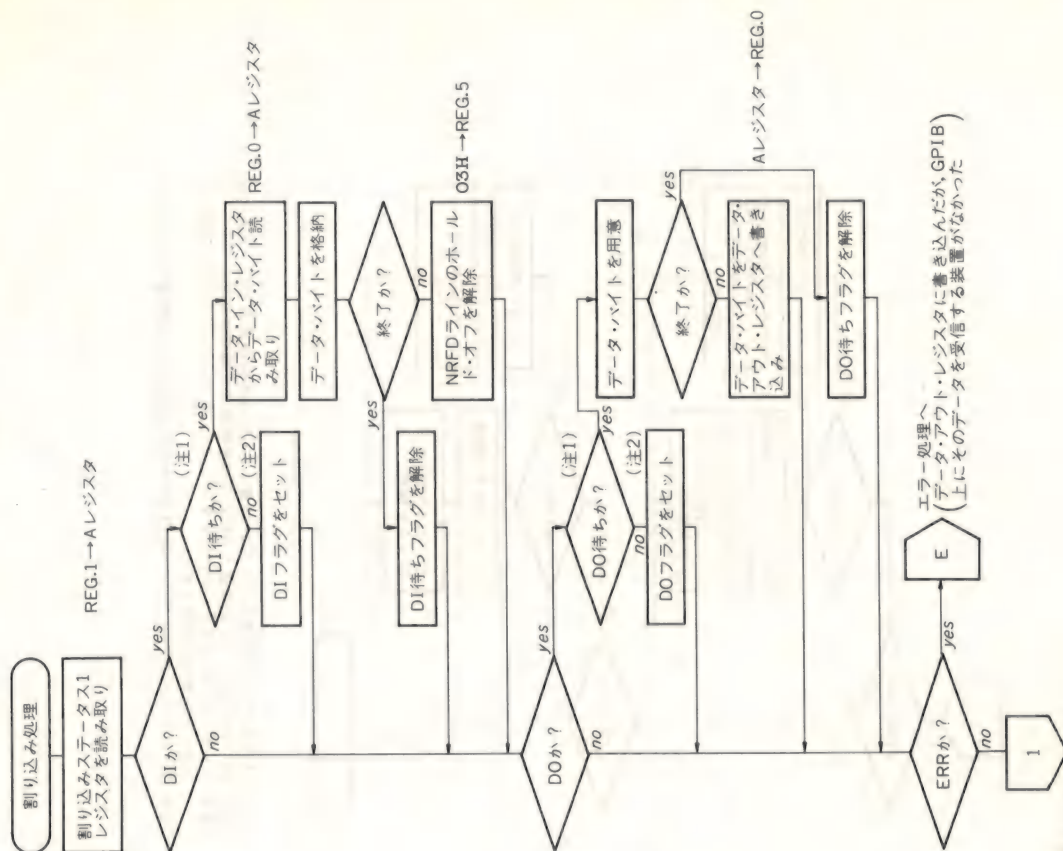
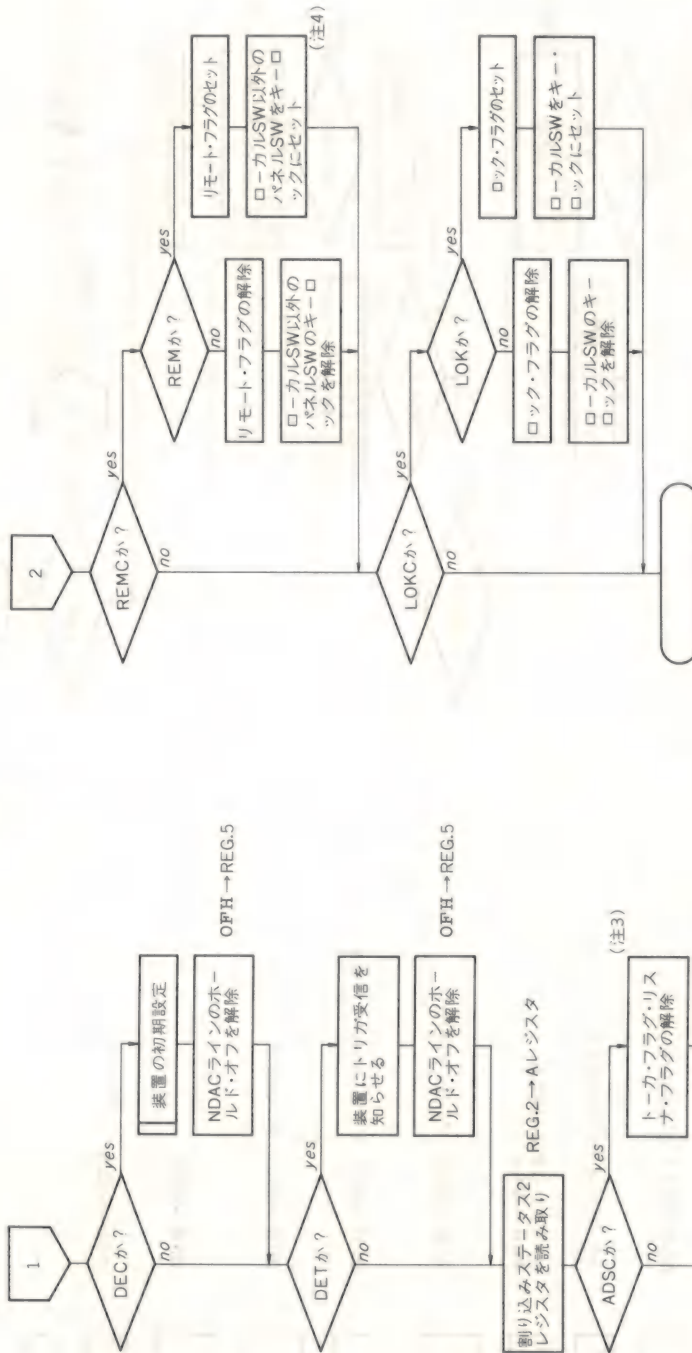
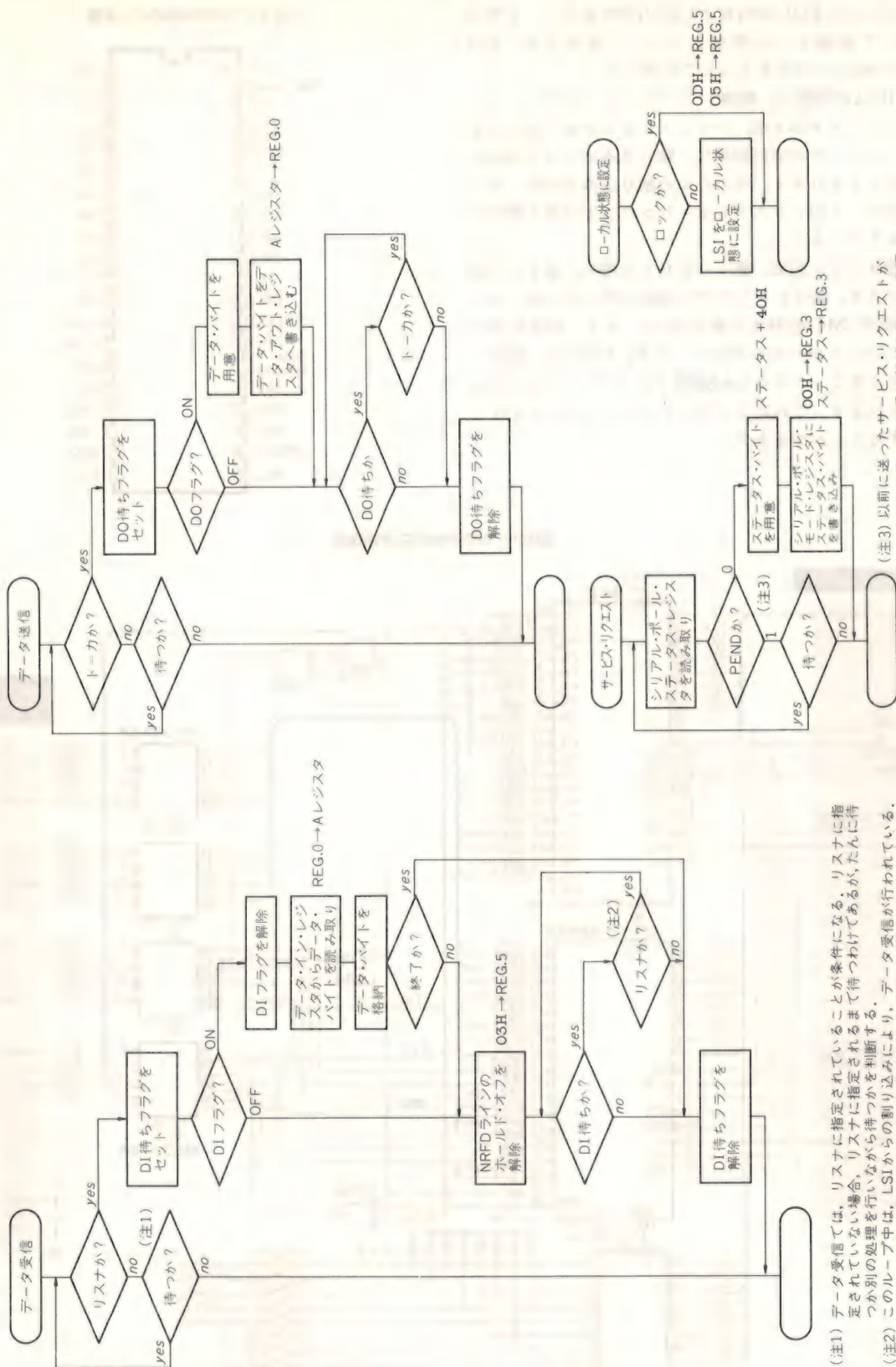


図 7> 割り込み処理フローチャート(つづき)



(注1) DI待ちフラグ、DO待ちフラグは、装置からLSIへの処理要求でセットされる。
 (注2) DIフラグ、DOフラグは装置からLSIへの処理要求で解除される。
 (注3) トーカー・フラグ、リスナー・フラグ、リモート・フラグ、ロック・フラグは、GPIBのステータス表示に利用できる。
 (注4) ローカルスイッチは、リモート状態をローカル状態に変える装置のスイッチ。

〈図8〉装置処理のフローチャート



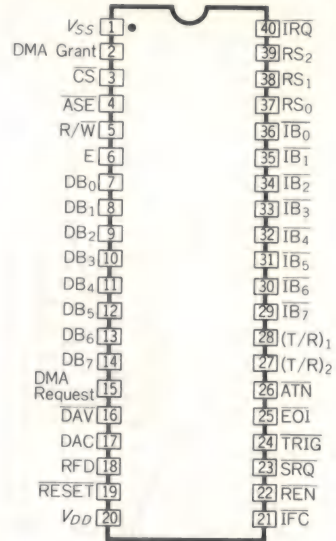
ATN, IFC, EOI, SRQおよびREN信号線をハードを付加して制御する必要があります。簡単には、PIA (MC6821)を付加することで可能です。

GPIAの制御は、制御レジスタによって行い、リード・レジスタが8個、ライト・レジスタが7個あります。コマンドの実行結果は、割り込みレジスタ(R0R)にセットされます。そのために割り込みを使用しない場合は、このレジスタをポーリングして状態を確認するようにします。

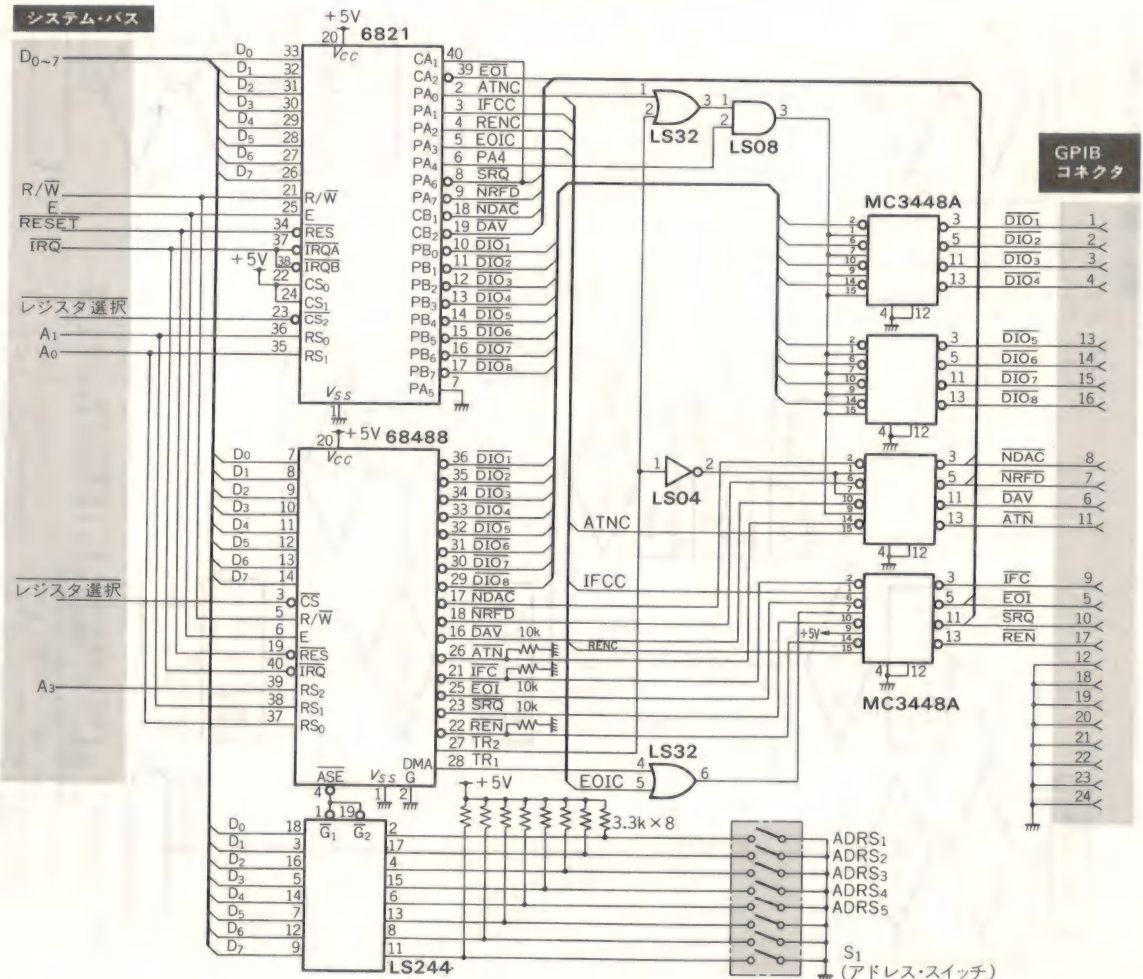
図9にピン配置、表1に各ピンの説明、表2に内部レジスタ、表3にレジスタの説明をそれぞれ示します。

図10にMC68488の回路図を示します。PIAを使用してコントロールも可能にします。GPIAは、R4RレジスタをリードするとASE端子から“L”パルスが出力されます。これによってアドレス・スイッチをリードすることができます。

〈図9〉⁽⁶⁾ MC68488のピン配置



〈図10〉⁽¹⁾ MC68488の応用回路図



◆ MC68488のソフトウェア

GPIAの制御ソフトは、初期設定、トーカ、リスナおよびコントローラについて説明します。今回は、割り込みを使用していません。

● 初期設定

電源投入時にハード・リセットが行われますが、制

御レジスタ(R3W)によってリセットします。以下処理手順を示します。

- ① アドレス・スイッチのリード
- ② アドレス・レジスタの設定
- ③ リセットのクリア
- ④ PIAの初期設定
- ⑤ トーカ、リスナの設定

〈表 1〉⁽⁶⁾ MC68488の各入出力ピン説明

ピン番号	名称	各 端 子 の 働 き	ピン番号	名称	各 端 子 の 働 き
1	V _{SS}	グラウンド端子	21	IFC	GPIAを既知状態にするクリア端子
2	DMA Grant	DMA要求を知らせる入力端子 使用しない時は必ず接地	22	REN	リモート“L”，ローカル“H”の選択端子
3	CS	GPIAをセレクトするチップ・セレクト端子	23	SRQ	サービス要求を出力する端子
4	ASE	アドレス設定の時使用する端子	24	TRIG	IEEE・標準バスからのGETコマンドを知らせる端子
5	R/W	レジスタのアクセスやデータの方向性を決める リード・ライト端子	25	EOI	データの出力転送の時、最後であることを知らせる端子
6	E	φ ₂ クロックの入力端子	26	ATN	データ・バス上の信号がデータ“H”であるのかコマンドある いはアドレス“L”であるかを示すアテンション端子
7	DB ₀	MPUとGPIA間でのデータ伝送端子	27	(T/R) ₂	トランシーバ(MC3448)のEOI以外のデータ伝送 の方向性を制御する端子
8	DB ₁		28	(T/R) ₁	トランシーバ(MC3448)のEOI信号の方向性を制 御する端子
9	DB ₂		29	IB ₇	GPIBとGPIA間でのデータ伝送端子
10	DB ₃		30	IB ₆	
11	DB ₄		31	IB ₅	
12	DB ₅		32	IB ₄	
13	DB ₆		33	IB ₃	
14	DB ₇		34	IB ₂	
15	DMA Request	DMAコントローラに対し、DMAリクエストを 出力する端子	35	IB ₁	
16	DAV	ハンドシェイク・ラインのひとつで、トーカの際 データが有効であることを示す	36	IB ₀	
17	DAC	ハンドシェイク・ラインのひとつで、リスナの際 データの受け取り可能な時、真になる	37	RS ₀	内蔵されている15個のレジスタのセレクト端子
18	RFD	ハンドシェイク・ラインのひとつで、リスナの際 データを受け取った時、真になる	38	RS ₁	
19	RESET	MPUなどからハード的にGPIAをリセットする端子	39	RS ₂	
20	V _{DD}	+5V端子	40	IRQ	割り込み要求の出力端子

〈表 2〉⁽⁶⁾ MC68488内部レジスタ

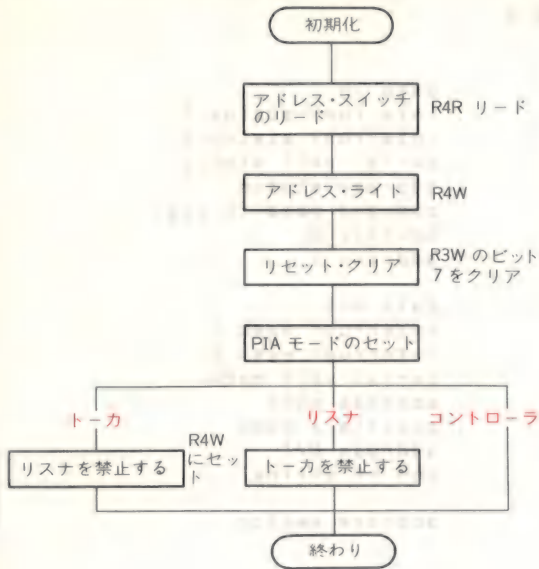
レジスタ名称	名称	RS ₂	RS ₁	RS ₀	R/W	各 レジスタ の 内 容							
						7	6	5	4	3	2	1	0
割り込みステータス	R0R	0	0	0	1	INT	BO	GET	—	APT	CMD	END	BI
割り込みマスク	R0W	0	0	0	0	IRQ	BO	GET	—	APT	CMD	END	BI
コマンド・ステータス	R1R	0	0	1	1	UACG	REM	LOK	—	RLC	SPAS	DCAS	UUCG
(未使用)	—	0	0	1	0	—	—	—	—	—	—	—	—
アドレス・ステータス	R2R	0	1	0	1	ma	to	lo	ATN	TACS	LACS	LPAS	TPAS
アドレス・モード	R2W	0	1	0	0	dsel	to	lo	—	hlde	hlda	—	apte
補助コマンド	R3R	0	1	1	1	RESET	DAC	DAV	RFD	msa	rtl	ulpa	fget
補助コマンド	R3W	0	1	1	0	RESET	rldr	feoi	dacr	msa	rtl	dacd	fget
アドレス・スイッチ	R4R	1	0	0	1	UD ₃	UD ₂	UD ₁	AD ₅	AD ₄	AD ₃	AD ₂	AD ₁
アドレス	R4W	1	0	0	0	lsbe	dal	dat	AD ₅	AD ₄	AD ₃	AD ₂	AD ₁
シリアル・ポール	R5R	1	0	1	1	S ₇	SRQS	S ₅	S ₄	S ₃	S ₂	S ₁	S ₀
シリアル・ポール	R5W	1	0	1	0	S ₇	rsv	S ₅	S ₄	S ₃	S ₂	S ₁	S ₀
コマンド・バススルー	R6R	1	1	0	1	B ₇	B ₆	B ₅	B ₄	B ₃	B ₂	B ₁	B ₀
パラレル・ポール	R6W	1	1	0	0	PPR ₈	PPR ₇	PPR ₆	PPR ₅	PPR ₄	PPR ₃	PPR ₂	PPR ₁
データ・イン	R7R	1	1	1	1	DI ₇	DI ₆	DI ₅	DI ₄	DI ₃	DI ₂	DI ₁	DI ₀
データ・アウト	R7W	1	1	1	0	DO ₇	DO ₆	DO ₅	DO ₄	DO ₃	DO ₂	DO ₁	DO ₀

〈表3〉⁽⁵⁾ MC68488の内部レジスタの説明

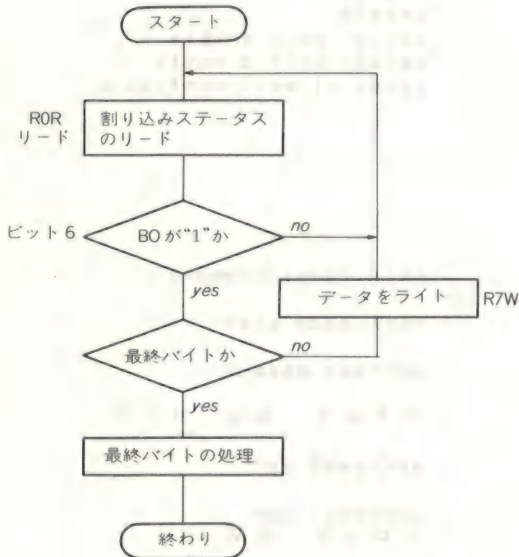
レジスタ名	ビット名	各 ビ ッ ト の 働 き	レジスタ名	ビット名	各 ビ ッ ト の 働 き
割り込み ステータス・ レジスタ (R0R)	BI	R7RにGPIBからのデータが読み込まれたかどうかの判断をする	補助コマ ンド・ レジスタ (R3W)	fget	TRIG端子を“H”にする時セットする
	END	EOIが“L”でATNが“H”の時セットされる		dacd	DACハンドシェイクを停止する時セットする
	CMD	コントロールから特別なコマンドが送信された時セットされる		rtl	セットするとリモートからローカル状態になる
	APT	R6Rから第2アドレスが読み込まれた時セットされる		msa	第2コマンド・グループの伝送が行われている時セットする
	GET	GETコマンドの判断をする		dacr	停止していたDACハンドシェイクを復帰する時セットする
	BO	R7Wに書き込みが行われたかどうかを判断する		feoi	EOI信号を出力する時セット
	INT	R0Wに書き込まれた内容により変化する		rfdr	RFDホールド・オフ・コマンドにより、停止していたハンドシェイクを復帰する時セット
割り込み マスク・ レジスタ (R0W)	BI	R0Rのそれぞれに対応する割り込みをマスクする	アドレス・ スイッチ・ レジスタ (R4R)	RESET	GPIAをリセットする時セットする
	END			AD ₁	GPIAのアドレス・スイッチのためのビット
	CMD			AD ₅	
	APT			UD ₁	ユーザが定義できるビット
	GET			UD ₃	
	BO			AD ₁	GPIAのアドレス書き込みのビット
	IRQ	IRQ端子からの割り込みをマスクする		AD ₅	
コマンド・ ステータス・ レジスタ (R1R)	UUCG	コントロールから×001××××(×は1または0)コマンドが送信されるとセットされる	アドレス・ レジスタ (R4W)	dat	トーク機能を停止するビット
	DCAS	デバイス・クリア・アクティブ状態の時セットされる		dal	リスナ機能を停止するビット
	SPAS	シリアル・ボール・アクティブ状態の時セットされる		lsbe	デュアル・プライマリ・アドレス・モードにするビット
	RLC	リモート/ローカル状態に対応して変化する		S ₀	シリアル・ボール・ を行う時のビット
	LOK	REN端子が“L”で、ローカル・ロックアウト・コマンド(×0010001)が送信されるとセットされる		S ₅	
	REM	リモート/ローカル状態を示すビット		S ₇	
	UACG	定義されていないアドレス・グループ・コマンドが送信されるとセットされる	シリアル・ ボール・ レジスタ (R5R/W)	SRQS(R)	rsvに書き込んだ内容を示す
アドレス・ ステータス・ レジスタ (R2R)	TPAS	第2アドレス・モード(apte=1)の時セットされ、第1トーク・アドレスを受信したことを示す		rsv(W)	サービス要求する場合セット
	LPAS	第2アドレス・モード(apte=1)の時セットされ、第1リスナ・アドレスを受信したことを示す	コマンド・ パススルー・ レジスタ (R6R)	B ₀	IEEE-488標準バスのデータ・ラインの情報を示すビット
	LACS	リスナ・アクティブ状態の時セットされる		B ₇	
	TACS	トーク・アクティブ状態の時セットされる	パラレル・ ボール・ レジスタ (R6W)	PPR ₁	パラレル・ボールを行うビット
	ATN	ATN端子の状態を示す		PPR ₈	
	lo	リスン・オンリ・モードの時セットされる	データ・ イン・ レジスタ (R7R)	DI ₀	GPIBからデータを受け取るビット
	to	トーク・オンリ・モードの時セットされる		DI ₇	
アドレス・ モード・ レジスタ (R2W)	ma	TACS, LACS, SPASになった時などにセットされる	データ・ アウト・ レジスタ (R7W)	DO ₀	MPUからデータをGPIBへ伝送するビット
	apte	拡張アドレッシング・モードまたは第2アドレッシング・モードにする時にセットする		DO ₇	
	hlde	リスナ・モードの時、RFDハンドシェイクをホールド・オフする時セット			
	hlde	リスナ・モードでかつ、EOI信号を受け取った時、RFDハンドシェイクをホールド・オフする時セット			
	lo	リスン・オンリ・モードにする時にセットする			
	to	トーク・オンリ・モードにする時にセットする			
	dsel	DCAS, UACG状態などにしない場合などにセットする			
補助コマ ンド・ レジスタ (R3R)	fget	TRIG端子の状態を知らせるビット			
	ulpa	第1アドレスを受け取った時、最初のビットの内容を示す			
	rtl	リモート状態からローカル状態になるとセットされる			
	msa	第2アドレスが正しい存在状態の時セットされる			
	RFD	RFD端子の状態を示すビット			
	DAV	DAV端子の状態を示すビット			
	DAC	DAC端子の状態を示すビット			
	RESET	リセット状態を示すビット			

(注) 各ビットの働きのすべてを記載したわけではない。
詳細は、GPIA ユーザ・マニュアルを参照。

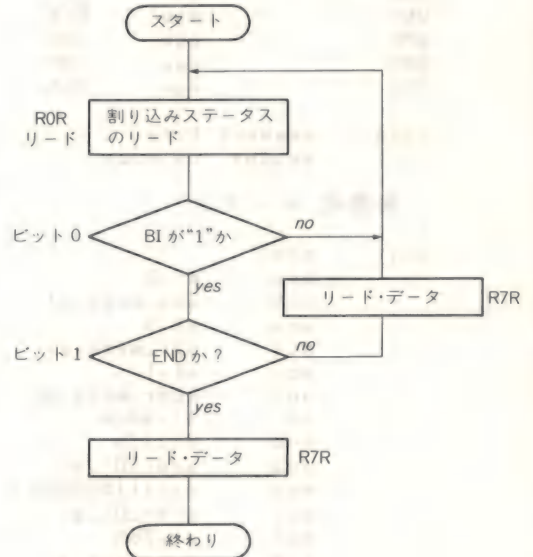
〈図11〉 MC68488初期化のフローチャート



〈図12〉⁽⁶⁾ トーカの処理



〈図13〉⁽⁶⁾ リスナの処理



などを行います。

図11に初期設定のフローチャートを示します。

● トーカの処理

トーカの場合は、データの送信を行います(図12のフローチャート参照)。

● リスナの処理

リスナの場合は、データの受信を行います(図13のフローチャート参照)。

● コントローラの処理

コントローラの場合は、その機能によりトーカ、リスナに対してサービスを行います。割り込みにより、トーカ、リスナのデータ送受信の処理を行います。

GPIB上にメッセージを送る場合、ATN線によってコマンドまたはデータの区別を行います。

```

; GLIP μPD7210 プログラム
;
;
data_in      equ      0                ; data in
int_sts1     equ      data_in+2        ; interrupt status 1
int_sts2     equ      int_sts1+2       ; interrupt status 2
spl_sts      equ      int_sts2+2       ; serial poll status
addr_sts     equ      spl_sts+2        ; address status
cmd_thr      equ      addr_sts+2       ; command pass through
addr0        equ      cmd_thr+2        ; address 0
addr1        equ      addr0+2          ; address 1
;
data_out     equ      data_in          ; data out
int_msk1     equ      int_sts1         ; interrupt mask 1
int_msk2     equ      int_sts2         ; interrupt mask 2
spl_mode     equ      spl_sts          ; serial poll mode
addr_mode    equ      addr_sts         ; address mode
aux_mode     equ      cmd_thr          ; auxiliary mode
addr_01      equ      addr0            ; address 0/1
end_str      equ      addr1            ; end of string
;
adsw         equ      addr1+2          ; address switch
cr           equ      0dh              ;
;
; command
;
UNL          equ      3fh              ; unlisten
UNT          equ      5fh              ; untalk
SPE          equ      18h              ; serial poll enable
SPD          equ      19h              ; serial poll disable
PPC          equ      05h              ; parallel poll configure
;
code         segment byte
            assume cs:code
;
; 初期化 ルーチン
;
init         proc
            mov     al,2                ; chip reset command
            out     aux_mode,al
            mov     al,0                ; poll mode clear
            out     spl_mode,al
            mov     al,1                ; address mode 1
            out     addr_mode,al
            in      al,adsw             ; アドレス sw リード
            and     al,1fh
            out     addr_01,al          ; address0 set
            mov     al,11100000b
            out     addr_01,al          ; address1 set
            mov     al,25h              ; クロック 5 M
            out     aux_mode,al
            mov     al,81h              ; RFD holdoff on all data mode
            out     aux_mode,al
            mov     al,0a0h             ; T1 = 2 μs
            out     aux_mode,al
            mov     al,0c3h             ;
            out     aux_mode,al
            mov     al,0
            out     aux_mode,al         ; chip on
            ret
init         endp
;
gpiib        proc
;
; コマンドの送信
;
; ES:BX = コマンドアドレス

```



```

; C X      = 送信 バイト数
;
cmd_send:
    call    set_casc                ; command mode set
cmd_out:
    mov     ah,es:[bx]              ; command byte
    call    cmd_put                 ; command output
    loop    cmd_out
    ret

; データの送信
;
; E S : B X = データアドレス
; C X      = 送信 バイト数
;
data_send:
    call    set_casc                ; command mode set
    mov     al,10h                  ; go to standby command
    out     aux_mode,al             ; data send mode
data_set:
    mov     ah,es:[bx]              ; command byte
    call    data_put                ; command output
    loop    data_set
    ret

; データの受信
;
; E S : B X = データアドレス
; C X      = 受信 バイト数
;
data_recv:
    call    recv_data               ; data input
    mov     es:[bx],al
    inc     cx
    cmp     al,CR                   ; end code ?
    je      recv_end
    in      al,addr1
    test    al,80h                  ; EOI ?
    jz      data_recv
recv_end:
    ret

; シリアル ポール
;
; D L = トーカアドレス
;
srll_poll:
    call    set_casc                ; command mode
    mov     ah,UNL                  ; Unlisten cmd.
    call    cmd_put
    mov     ah,SPE                  ; serial poll enable cmd.
    call    cmd_put
next_poll:
    mov     ah,dl                   ; トーカアドレス send
    call    cmd_put
    mov     al,10h
    out     aux_mode,al             ; go to standby cmd.
    call    recv_data               ; STB read
    push    ax
    call    set_casc                ; command mode
    pop     ax
    test    al,40h                  ; REQ on ?
    jz      next_poll
    push    ax
    mov     ah,SPD
    call    cmd_put                 ; serial poll disable

```

```

mov     ah,UNT
call    cmd_put           ; untalk cmd.
pop     ax
ret

;
; パラレル ポール
;
; DL = リスナアドレス
; DH = 2次アドレス PPE または PPD
;
pri_poll:
call    set_casc
mov     ah,UNL
call    cmd_put           ; UNL cmd send
mov     ah,dI
call    cmd_put           ; リスナアドレス
mov     ah,PPC
call    cmd_put           ; PPC send
mov     ah,dh
call    cmd_put           ; PPE or PPD
mov     al,ldh
out     aux_mode,al      ; execute parallel poll
poll_chk:
in      al,int_sts2
test    al,10h           ; co=1
jz      poll_chk         ; no
in      al,cmd_thr       ; poll response get
push    ax
mov     ah,UNL
call    cmd_put
pop     ax
ret

; command send mode
;
set_casc:
in      al,addr_sts      ; address status read
test    al,40h           ; ATN ?
jnz     casc_exit
mov     al,12h           ; take control sync command
out     aux_mode,al
casc_exit:
ret

;
cmd_put:
in      al,int_sts2      ; interrupt status 2
test    al,8h           ; command out ?
jz      cmd_put         ; no
mov     al,ah
out     data_out,al      ; command out
ret

;
data_put:
in      al,int_sts2      ; interrupt status 2
test    al,2h           ; data out ?
jz      data_put         ; no
mov     al,ah
out     data_out,al      ; data out
ret

;
recv_data:
in      al,int_sts2
test    al,1            ; data in ?
jz      recv_data       ; no
in      al,data_in
ret

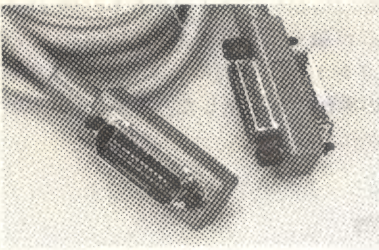
gpiib   endp
code    ends
end

```


§ 3-3

GPIB ラインを光ファイバで 延長する

鶴野和孝



最近、FA化の波にのって、GPIBを標準またはオプションで装備した計測器やパソコンが多くなりました。

一方、オプト・エレクトロニクスの発達により、ゲートICなみに簡単に取り扱える光ファイバ・リンク・モジュールが各社より発表されています。この光ファイバ・リンクを利用することにより、

- ① 電磁誘導ノイズを受けない、あるいは出さない。
- ② 装置間がアイソレートされるので、短絡の危険がない。

などの利点が得られ、これらをGPIBと組み合わせて考えると、

- ① GPIBを規格(最大20m)以上に延長できる。
- ② アナログ系に混入するノイズを低減できる。
- ③ 機器の故障がほかへ波及しない。

などの効果をあげられます。

そこで、ここでは広い工場や研究所などで計測システムを構築することを念頭におき、GPIBを規格以上に延長するという点について、製作実験を行った結果

を報告したいと思います。

● 光ファイバ・リンク

各社より発表されている光ファイバ・リンクの中から、必要な周辺回路までがモールド・パッケージ化されているTOML75(東芝)を使用しました。

光ファイバ・リンクTOML75⁽¹⁾は、写真1のように光送受信モジュールTODX75、光コネクタTOCP75、プラスチック光ファイバ(石英製もある)から構成されています。それぞれの電気的、光学的特性を表1に示



〈写真1〉光モジュールとコネクタ

〈表1(a)〉
トスリンクの絶対最大定格

型 名	項 目		記号	最小	最 大	単位
TOML70/TOML75	保 存 温 度		T_{stg}	-40	75	℃
	動 作 温 度		T_{opr}	0	70	℃
TOTX70	供 給 電 圧		V_{CC}	-0.5	7	V
	入 力 電 圧		V_{IN}	-0.5	5.5	V
TORX70	供 給 電 圧		V_{CC}	-0.5	7	V
	低レベル出力電流		I_{OL}	—	20	mA
	高レベル出力電流		I_{OH}	—	-1	mA
TODX75	供 給 電 圧		V_{CC}	-0.5	7	V
	入 力 電 圧		V_{IN}	-0.5	5.5	V
	低レベル出力電流		I_{OL}	—	20	mA
	高レベル出力電流		I_{OH}	—	-1	mA
TOCP70, TOCP70P TOCP75, TOCP75P	張力	光コネクタ/光ファイバ	T_{CF}	—	5	N
		光ファイバ	T_F	—	50(100) ⁽¹⁾	N
		光ファイバ曲げ半径		r	15	—
TOCP70Q, TOCP70R, TOCP70S TOCP75Q, TOCP75R, TOCP75S	張力	光コネクタ/光ファイバ	T_{CF}	—	50	N
		光ファイバ	T_F	—	250(1000) ⁽²⁾	N
		光ファイバ曲げ半径		r	50	—

注(1) ()内は TOCP70P, TOCP75P の値

注(2) ()内は TOCP75R, TOCP75S の値

します。

バス延長装置の設計

GPIBでは、とくに各機器の応答速度を制限していません。したがって、§3-1の図3 (p.77)における④

～⑩までのどの区間をも引き延ばすことができます。例えば、機器の代わりにスイッチやLEDをバスに接続し、LEDの点滅を見ながら人間がスイッチのON/OFFでハンドシェイクを進めることもできます。この応答速度の柔軟性を利用した、簡易なGPIBのデバッグも発表されています。

〈表1(b)〉 トスリンクの電氣的/光学的特性

($T_{opr}=25^{\circ}\text{C}$, $V_{CC}=5\pm0.25\text{V}$, $\lambda_P=660\text{nm}$)

型 名	項 目	記号	最小	標準	最大	単位	条 件
TOML70	伝 送 容 量		DC	—		Mbit/sec	任意符号 ⁽¹⁾
	伝搬遅延時間("L"→"H")	t_{PLH}	—	175	250	ns	ファイバ長1m, Tx入力とRx出力間
TOML75	伝搬遅延時間("H"→"L")	t_{PHL}	—	175	250	ns	ファイバ長1m, Tx入力とRx出力間
TOTX70	ファイバ結合光出力(1)	P_f	-15	-13	-10	dBm	外付け抵抗150Ω, TOCP70-1MBを介して ⁽²⁾
	ファイバ結合光出力(2)	P_f	-20	-17	-14	dBm	外付け抵抗150Ω, TOCP70Q-1MBを介して ⁽²⁾
	ピーク発光波長	λ_P	—	660	—	nm	
	消費電流	I_{CC}	—	45	65	mA	
	高レベル入力電圧	V_{IH}	2.0	—	—	V	⁽³⁾
	低レベル入力電圧	V_{IL}	—	—	0.8	V	⁽³⁾
	高レベル入力電流	I_{IH}	—	—	40	μA	$V_{CC}=5.25\text{V}$, $V_{IH}=2.4\text{V}$
	低レベル入力電流	I_{IL}	—	—	-1.6	mA	$V_{CC}=5.25\text{V}$, $V_{IL}=0.4\text{V}$
TORX70	最大受信電力	P_{MAX}	-15	-13	—	dBm	$BER=10^{-9}$ ⁽²⁾
	最小受信電力	P_{MIN}	—	-30	-28	dBm	$BER=10^{-9}$ ⁽²⁾
	消費電流	I_{CC}	—	18	30	mA	
	高レベル出力電圧	V_{OH}	4.6	—	—	V	$V_{CC}=4.75\text{V}$, $I_{OH}=-60\mu\text{A}$ ⁽⁵⁾ , $-50\mu\text{A}$ ⁽⁴⁾⁽⁶⁾
	低レベル出力電圧(1)	V_{OL}	—	—	0.4	V	$V_{CC}=4.75\text{V}$, $I_{OL}=1.2\text{mA}$ ⁽⁴⁾⁽⁵⁾
	低レベル出力電圧(2)	V_{OL}	—	—	0.5	V	$V_{CC}=4.75\text{V}$, $I_{OL}=2\text{mA}$ ⁽⁴⁾⁽⁶⁾
TODX75	ファイバ結合光出力(1)	P_f	-15	-13	-10	dBm	外付け抵抗150Ω, TOPC75-1MBを介して
	ファイバ結合光出力(2)	P_f	-20	-17	-14	dBm	外付け抵抗150Ω, TOPC75Q-1MBを介して
	ピーク発光波長	λ_P	—	660	—	nm	
	高レベル入力電圧	V_{IH}	2.0	—	—	V	⁽³⁾
	低レベル入力電圧	V_{IL}	—	—	0.8	V	⁽³⁾
	高レベル入力電流	I_{IH}	—	—	40	μA	$V_{CC}=5.25\text{V}$, $V_{IH}=2.4\text{V}$
	低レベル入力電流	I_{IL}	—	—	-16	mA	$V_{CC}=5.25\text{V}$, $V_{IL}=0.4\text{V}$
	最大受信電力	P_{MAX}	-15	-13	—	dBm	$BER=10^{-9}$ ⁽²⁾
	最小受信電力	P_{MIN}	—	-30	-28	dBm	$BER=10^{-9}$ ⁽²⁾
	消費電流	I_{CC}	—	63	95	mA	
	高レベル出力電圧	V_{OH}	4.6	—	—	V	$V_{CC}=4.75\text{V}$, $I_{OH}=-60\mu\text{A}$ ⁽⁵⁾ , $-50\mu\text{A}$ ⁽⁴⁾⁽⁶⁾
	低レベル出力電圧(1)	V_{OL}	—	—	0.4	V	$V_{CC}=4.75\text{V}$, $I_{OL}=1.2\text{mA}$ ⁽⁴⁾⁽⁵⁾
	低レベル出力電圧(2)	V_{OL}	—	—	0.5	V	$V_{CC}=4.75\text{V}$, $I_{OL}=2\text{mA}$ ⁽⁴⁾⁽⁶⁾
プラスチック・ファイバ	開 口 数	NA	—	0.5	—		
	屈折率分布	ステップ・インデックス					
	伝送損失		—	0.4	—	dB/m	ファイバ長10mで測定
	帯域幅		—	60	—	MHz	ファイバ長50m
	伝搬遅延時間	t_P	—	5	7	ns/m	
ポリマ・クラッド 石英コア・ファイバ	開 口 数	NA	—	0.41	—		
	屈折率分布	ステップ・インデックス					
	伝送損失		—	0.025	—	dB/m	ファイバ長100mで測定
	帯域幅		—	40	—	MHz	ファイバ長500m
	伝搬遅延時間	t_P	—	5	7	ns/m	

注(1) NRZ 符号の場合は3Mbit/secまで動作

注(2) ピーク値で規定

注(3) 高レベル入力時: 光出力 OFF, 低レベル入力時: 光出力 ON

注(4) 光入力 OFF 時: 高レベル出力, 光入力 ON 時: 低レベル出力

注(5) LS TTL を3個接続した場合

注(6) S TTL を1個接続した場合

ところで、GPIBを利用したシステムを構築する場合、1台の機器でも20m以上の距離が離れていると規格を満足できません。

図1で示すような接続を行った場合、転送データの信頼性から装置間のケーブルの長さは、2m以下にする必要があります。またGPIBの負荷は、図2に示すように、3kと6.2kΩの分散方式となっているために、伝送速度には限界があります。またドライブ能力も48mAであるために、ケーブルを長くすることはノイズ・マージンを低下させます。そこで、基本的に図3のような場合を想定し、GPIBの延長装置を製作することにしました。

● GPIB延長装置の構成

GPIB延長装置の構成としてなるべく小型にするため、GPIBとのバッファ、光ファイバ・リンク・モジュール、UART、そしてZ80 CPUとメモリという構成にし、ソフトウェアによるハンドシェイクに依存することにしました。

図4にシステムのブロック図を示します。

● ATN応答対策

規格によれば、「ATN」がONした場合、すべての機器はそれまでのハンドシェイクを中断し、200ns以

内にそれに応答しなくてはなりません。これはソフトウェアでは追いつけませんので、図5のようにハードウェアでカバーします。ここでは、装置自身が出力していないのに「ATN」がONになればフリップフロップをセットし、「NRFD」をONにします。

ソフトウェアのほうは、ポーリングによって「ATN」のONを検出した後、「NRFD」と「NDAC」をONにします。そして、次に「NRFD」のみOFFにすると、同時に「ATN」ONをもう一方の延長装置に光信号で送ります。

● IFC応答対策

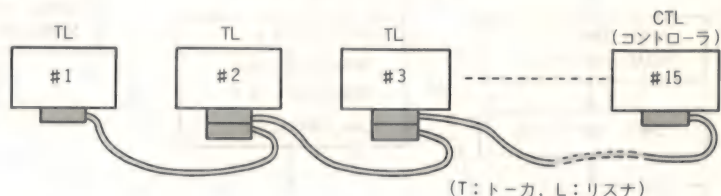
規格によると、「IFC」は100μs以上のパルスということになっています。そこで、これもいったんフリップフロップで受け、ポーリングで検出した後(図6)、もう一方の延長装置に光信号で送ります。

IFCの光信号を受信した延長装置では、ワンショットをトリガして、GPIB上に100μs以上のパルスを出すようにします。

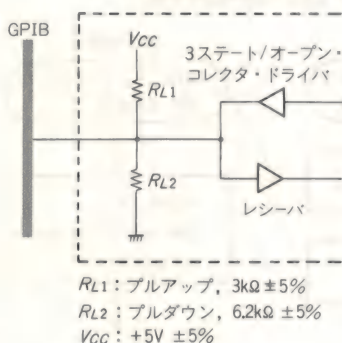
ソフトウェア

このGPIBの延長装置は、一方のバスの信号状態を、

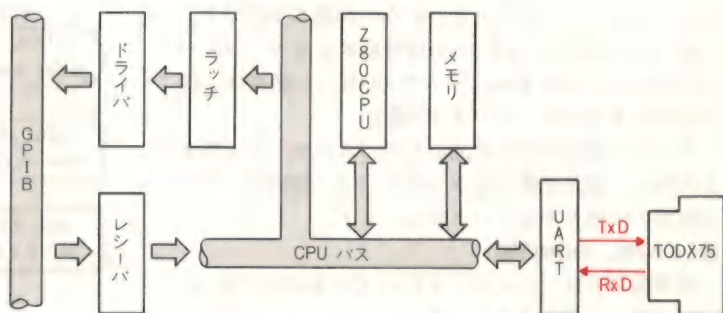
〈図1〉 GPIBの接続例



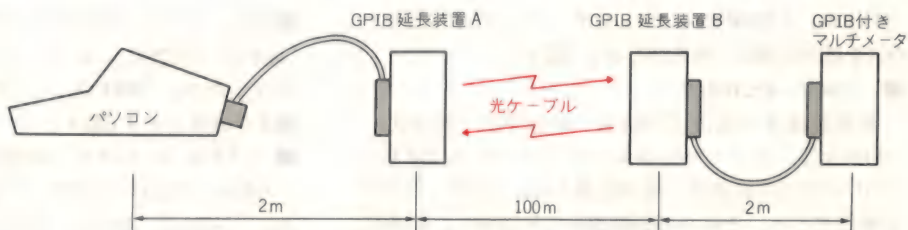
〈図2〉 GPIBの入出力部の特性



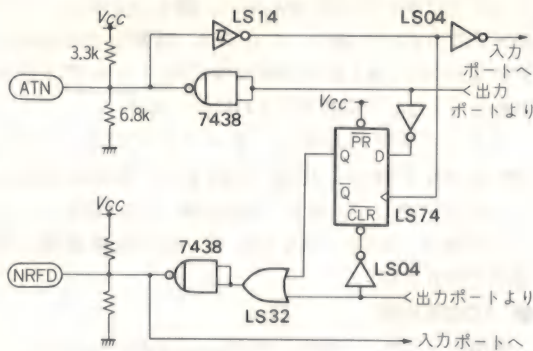
〈図4〉 GPIB延長装置のブロック図



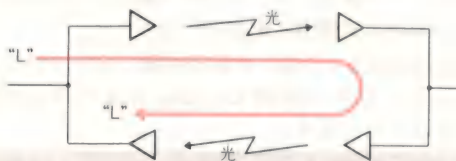
〈図3〉 GPIBインターフェースで20m以上延長する例



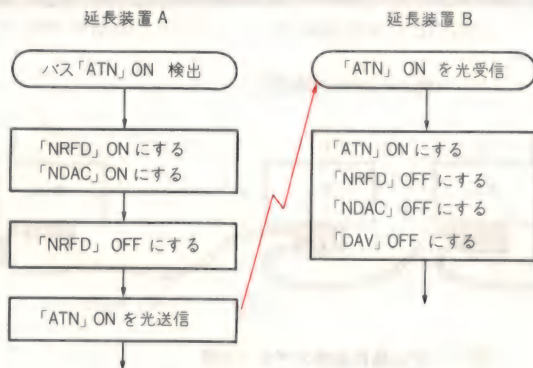
〈図5〉 ATNのタイミング発生回路



〈図7〉 信号の送受信の注意点



〈図8〉 ATNのON手順



もう一方のバスへ出力するというのが基本手法です。しかし、たんに一方のバスの信号状態をもう一方のバスへ出力しただけでは、いったんONした信号は二度とOFFにもどれません(図7)。

そこで、特定の信号に注目してバスのデータの向きを判断し、延長装置はもう一方のバスに接続されている機器の代理をするという形にします。

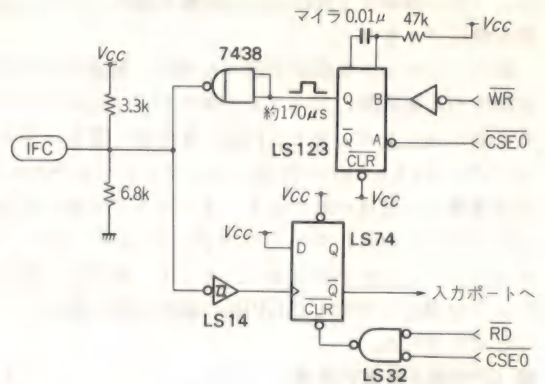
● 「ATN」のON応答

延長装置Aは、バスの「ATN」ONを検出すると、「NRFD」と「NDAC」をいったんONにし、「NRFD」はOFFにもどします。そして、「ATN」ONを延長装置Bに光送信します(図8)。

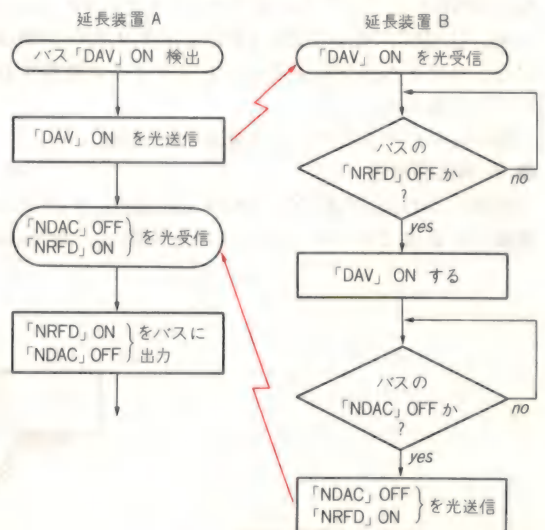
● 「DAV」のON応答

延長装置Aの出力「NRFD」がOFFで「NDAC」がONなら、コントローラまたはトーカーから「DAV」のONが出力されます。延長装置Aは、「DAV」のONを検出すると、それを延長装置Bに光送信し、延長装

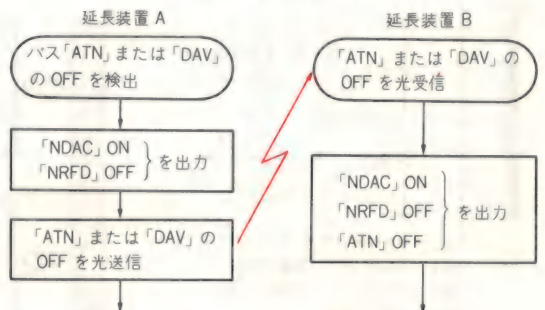
〈図6〉 IFCのタイミング発生回路



〈図9〉 DAVのON手順



〈図10〉 ATN, DAVのOFF手順

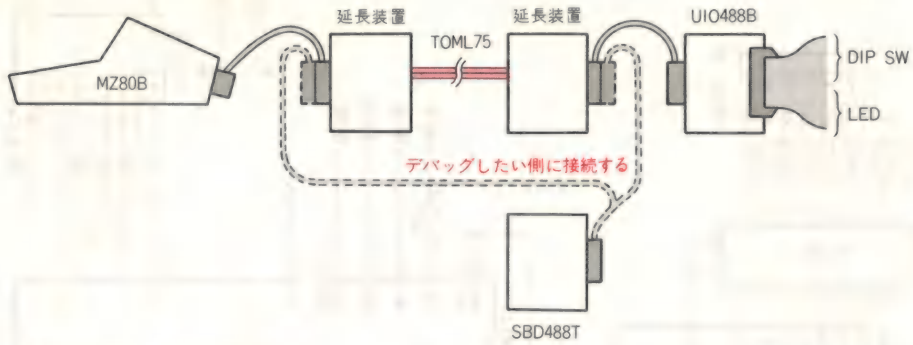


置Bは、バスの「NRFD」がOFFになるのを待って、「DAV」をONにします。そして、「NDAC」がOFFになったら、「NRFD」と「NDAC」の状態を延長装置Aに送信します(図9)。

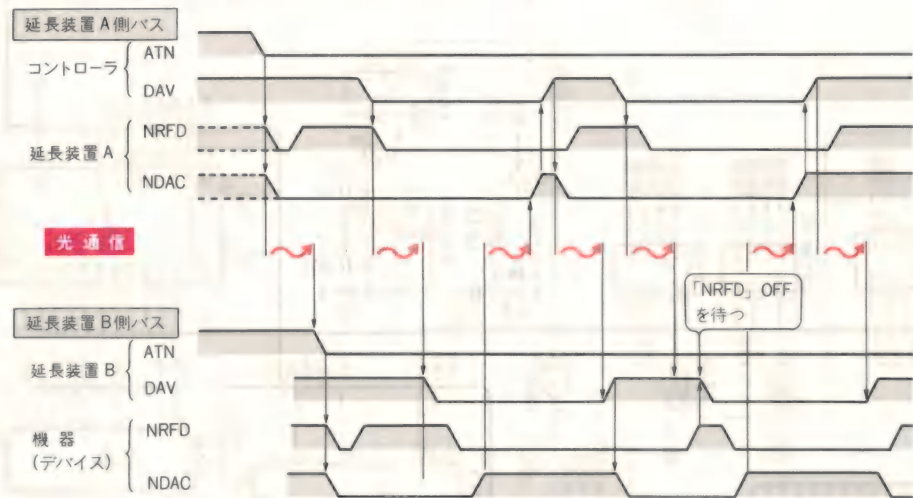
● 「ATN」と「DAV」OFF応答

「ATN」または「DAV」のOFFに対して、延長装置Aは「NDAC」のONと「NRFD」のOFFを出力し、

〈図11〉動作テストのブロック図



〈図13〉動作タイミング



次の「DAV」のONに備えます。また、「ATN」や「DAV」のOFFを延長装置Bに光送信します。

延長装置Bは、光受信すると「ATN」や「DAV」をOFFにし、「NDAC」のONと「NRFD」のOFFを出力します。この状態は、延長装置A、Bともに同じ出力です(図10)。

動作テスト

さて、実験に使用した装置は図11に示すように、コントローラとしてMZ80B(シャープ)、機器(デバイス)としてUIO488B(マイクロサイエンス)、さらにデバッグ用にSBD488T(マイクロサイエンス)を使用しました。

UIO488Bは、端末用の GPIB インターフェース・ユニットで、入出力ポート各 1 個を備えています。また、この UIO488B が、リスナに指定されているか、トーカーに指定されているかがわかるようになっているので、

これらのピンに LED やディップ・スイッチを接続し、動作が目で確認できるようにしました。

● 伝送速度

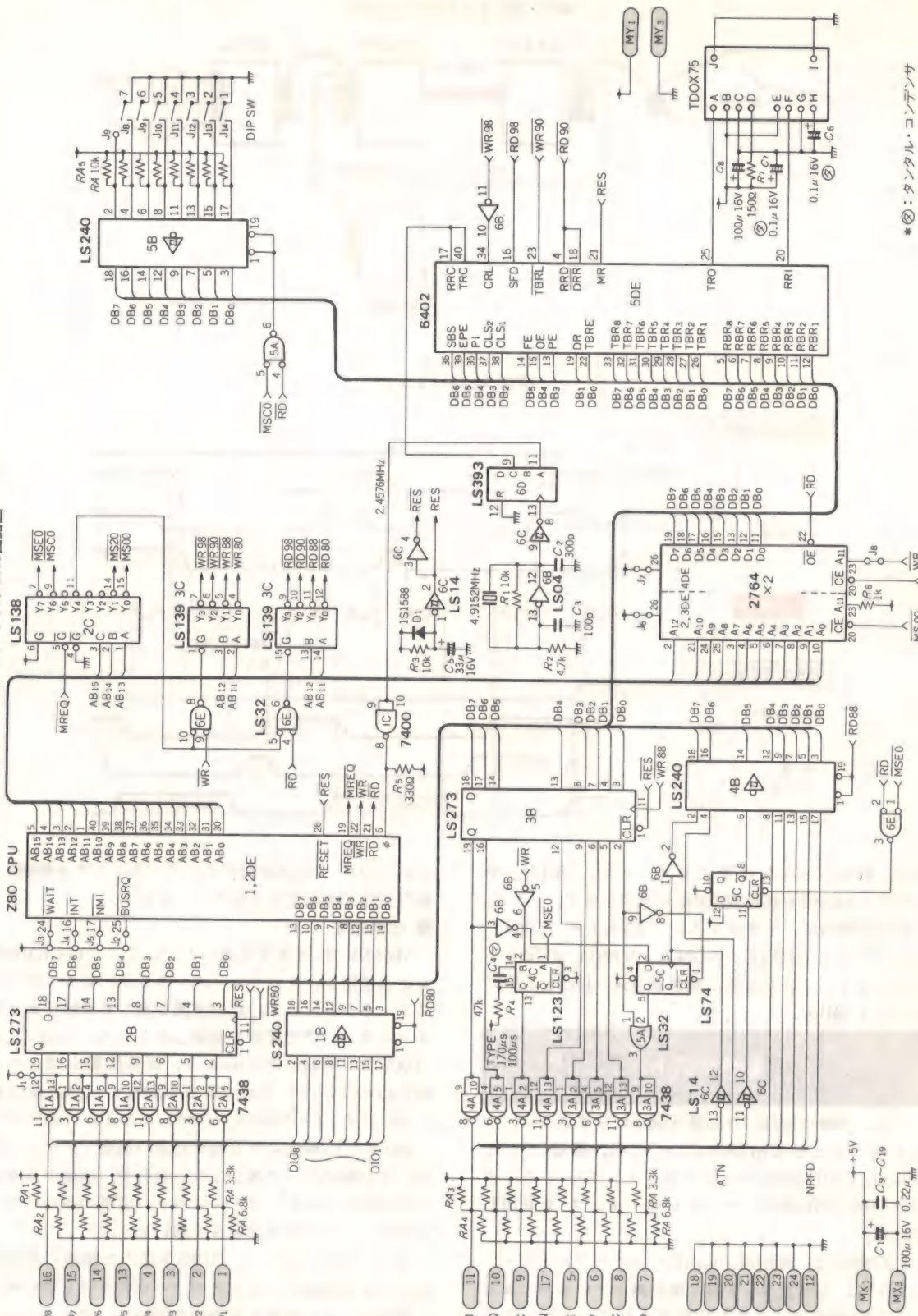
MZ80B の転送速度は遅いので、ここでは UIO488B を 2 台使用し、1 台をトーカー・オンリ、もう 1 台をリスナ・オンリにして、延長装置を通して実験しました。オシロスコープで波形を観測したところ、「DAV」と「DAV」の間隔が約 5 ms あり、延長装置を通さない場合は約 4 μ s でしたので、延長装置の伝送速度は約 5 ms/バイト (=200 バイト/秒)とみてよいと思います。

200 バイト/秒というと相当に遅い速度ですが、1 回当たりの転送データ数が少ない場合や、温度や真空などの信号のように、フィードバック応答の遅いシステムには、十分に応用できると思います。

また、何にも増して、GPIB を備えた機器に何の改造もせず遠隔操作できるメリットは大きいと思います。

最後に、この装置の回路図を図12に、タイミングを図13に、プログラムをリスト 1 に示します。

〈図12〉光インターフェース・システムの回路図



* ㊦: タンタル・コンデンサ
㊧: マイカ・コンデンサ

＜リスト1＞コントロール・プログラム

[illegible]

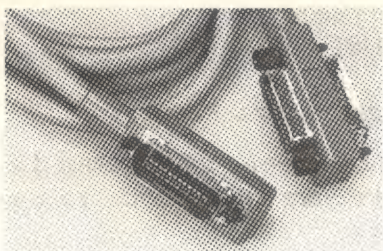
0076 7C	MOV	A,H	00C8 E6FE	ANI	1111110B ;NRFD = OFF
0077 A7	ANA	A	00CA 320320	STA	OC0BF
0078 C28000	JNZ	EDGE	00CD 320088	STA	OCB
007B 7D	MOV	A,L			
007C A7	ANA	A	00D0 3A0820	LDA	SC0BF
007D C8401	JZ	SEND ;SKIP IF NO EDGE	00D3 E6D0	ANI	1101000B ;IFC,E01,DAV,NDAC,NRFD = OFF
			00D5 320820	STA	SC0BF
	EDGE:		00D8 C38401	JMP	SEND
0080 7D	MOV	A,L ;RECALL ON EDGE WITHOUT SELF ON			
0081 E620	ANI	00100000B			
0083 C24301	JNZ	ONIFC			
0086 3E01	MVI	A,1		ONDAV:	
0088 320A20	STA	SNDBF	00DB 3A0080	LDA	ODB
008B 7D	MOV	A,L	00DE 320920	STA	SD0BF
008C E680	ANI	10000000B	00E1 3A0820	LDA	SC0BF
008E C2BE00	JNZ	ONATN	00E4 E6DC	ANI	1101100B ;IFC,NDAC,NRFD = OFF
0091 7D	MOV	A,L	00E6 320820	STA	SC0BF
0092 E604	ANI	00000100B	00E9 C38401	JMP	SEND
0094 C2DB00	JNZ	ONDAV			
0097 7C	MOV	A,H ;RECALL OFF EDGE			
0098 E680	ANI	10000000B		OFATN:	
009A C2EC00	JNZ	OFATN	00EC 3A0320	LDA	OC0BF
009D 7C	MOV	A,H	00EF E6FE	ANI	1111110B ;NRFD = OFF
009E E604	ANI	00000100B	00F1 F602	ORI	0000010B ;NDAC = ON
00A0 C2EC00	JNZ	OFDAV	00F3 320320	STA	OC0BF
00A3 7C	MOV	A,H	00F6 320088	STA	OCB
00A4 E602	ANI	00000010B	00F9 3A0820	LDA	SC0BF
00A6 C22401	JNZ	OFNDAC	00FC E6FC	ANI	1111100B ;NDAC,NRFD = OFF
00A9 7D	MOV	A,L	00FE F602	ORI	00000010B ;NDAC = ON
00AA E602	ANI	00000010B	0100 320820	STA	SC0BF
00AC C20601	JNZ	ONNDAC	0103 C38401	JMP	SEND
00AF 7D	MOV	A,L			
00B0 E601	ANI	00000001B	0106 3A0220	LDA	OCIBF
00B2 C21D01	JNZ	ONNRFD	0109 E601	ANI	00000001B
00B5 7C	MOV	A,H	010B C8401	JZ	SEND
00B6 E601	ANI	00000001B	010E AF	XRA	A
00B8 C21D01	JNZ	OFNRFD	010F 320A20	STA	SNDBF
00BB C38401	JMP	SEND	0112 3A0220	LDA	OCIBF
			0115 E6FD	ANI	1111101B
			0117 320220	STA	OCIBF
			011A C38401	JMP	SEND
	ONATN:				
00BE 3A0320	LDA	OC0BF		ONNRFD:	
00C1 E640	ANI	01000000B		OFNRFD:	
00C3 F603	ORI	00000011B			
00C5 320088	STA	OCB	011D AF	XRA	A

〈リスト1〉コントロール・プログラム(つづき)

011E 320A20	STA ;	SNDBF	017E 3A0090	LDA ;	SIOD	01DE AF	XRA STA ;	A RCTMR
0121 C38401	JMP ;	SEND	0181 C33300	JMP ;	MAIN	01DF 320120	;	
	OFNDAC: ;			SEND: ;		RSV2: ;		
0124 3A0820	LDA	SC0BF	0184 3A0020	LDA ;	TSTMR	01E2 3A0098	LDA	SI0C
0127 F601	ORI	00000001B	0187 FEFF	CF1	OFFH	01E5 E602	ANI	2
0129 320820	STA	SC0BF	0189 C8D01	JZ	#+4	01E7 C2ED01	JNZ	RSV24
012C 3A0220	LDA	SC0BF	018C 3C	INR	A	01EA C33300	JMP	MAIN
012F E601	ANI	00000001B	018D 320020	STA	TSTMR	RSV24: ;		
0131 C28401	JNZ	SEND	018F FE15	CF1	21	01ED 3A0090	LDA	SI0D
	;		0190 FE15	JC	RSV	01FO 320620	STA	SDIBF
0134 AF	XRA	A	0192 DABF01	;		SEEDGE: ;		
0135 320A20	STA	SNDBF	0195 3A0098	LDA	SI0C	01F3 3A0720	LDA	SCIBF
	;		0198 E601	ANI	1	01F6 4F	MOV	C,A
0138 3A0220	LDA	OCIBF	019A CABF01	JZ	RSV	01F7 3A0320	LDA	OC0BF
013B F602	ORI	00000010B		;		01FA 47	MOV	B,A
013D 320220	STA	OCIBF	019D 3A0A20	LDA	SNDBF	;	;	
	;		01A0 A7	ANA	A	01FB A9	XRA	C
0140 C38401	JMP	SEND	01A1 CABF01	JZ	RSV	01FC 5F	MOV	E,A
	;			;		01FD A1	ANA	C
	ONIFC: ;		01A4 3A0820	LDA	SC0BF	01FE 6F	MOV	L,A ;SAVE ON EDGE
	;		01A7 320090	STA	SI0D	;	;	
0143 AF	XRA	A		;		01FF 7B	MOV	A,E
0144 320220	STA	OCIBF	01AA 3A0098	LDA	SI0C	0200 A0	ANA	B
0147 320320	STA	OC0BF	01AD E601	ANI	1	0201 67	MOV	H,A ;SAVE OFF EDGE
014A 320520	STA	LOCIBF	01AF CAAA01	JZ	#-5	;	;	
014D 320720	STA	SCIBF		;		0202 E6DC	ANI	11011100B
0150 320820	STA	SC0BF	01B2 3A0920	LDA	SD0BF	0204 2F	CMA	
	;		01B5 320090	STA	SI0D	0205 57	MOV	D,A
0153 320080	STA	ODB		;		0206 3A0220	LDA	OCIBF
0156 320088	STA	OCB		;		0209 A2	ANA	D
0159 3A00E0	LDA	IFC ;CLEAR IFC LATCH	01B8 AF	XRA	A	020A 320220	STA	OCIBF
	;		01B9 320020	STA	TSTMR	;	;	
015C 3A0098	LDA	SI0C	01BC 320A20	STA	SNDBF	020D 7D	MOV	A,L
015F E601	ANI	1		;		020E E620	ANI	00100000B
0161 CASC01	JZ	#-5		RSV: ;		0210 C2D702	JNZ	RONIFC
0164 3E20	MVI	A,00100000B ;SET IFC	01BF 3A0120	LDA	RCTMR	;	;	
0166 320090	STA	SI0D	01C2 FEFF	CF1	OFFH	0213 7D	MOV	A,L
	;		01C4 CAC801	JZ	#+4	0214 E680	ANI	10000000B
0169 3A0098	LDA	SI0C	01C7 3C	INR	A	0216 C2A002	JNZ	RONATN
016C E601	ANI	1	01C8 320120	STA	RCTMR	;	;	
016E CA6901	JZ	#-5	01CB FE04	CF1	4	0219 7D	MOV	A,L
	;		01CD DAE201	JC	RSV2	021A E604	ANI	00000100B
	;			;		021C C25C02	JNZ	RONDAV
0171 AF	XRA	A		RSV1: ;		021F 7C	MOV	A,H
0172 320090	STA	SI0D	01D0 3A0098	LDA	SI0C	0220 E680	ANI	10000000B
	;		01D3 E602	ANI	2	0222 C2BE02	JNZ	ROFATN
0175 3EC8	MVI	A,200	01D5 C33300	JZ	MAIN	;	;	
0177 3D	DCR	A		;		0225 7C	MOV	A,H
0178 C27701	JNZ	#-1	01D8 3A0090	LDA	SI0D	0226 E604	ANI	00000100B
	;		01DB 320720	STA	SCIBF			
017B 3A0098	LDA	SI0C		;				

トランジスタ技術
SPECIAL

099A CLK	C000 DIP5W	0080 EDGE	E000 IFC	0000 INIT
A000 INTP	2005 LOC1BF	0033 MAIN	8800 DCB	2002 OC1BF
2003 CF0DBF	8000 ODB	2004 O0CBF	00EC OFATN	00EC OF0AV
0124 OFNDBF	011D ONFRD	00BE ONATN	00DB ONDAV	0143 ON1FC
0106 ONNDAC	011D ONNRF	2000 RANTOP	2001 ACTMR	02BE ROFATN
02BE ROF0AV	0000 R0MTP	0240 RONATN	0256 RONDAV	02BE R0NDV2
028C RONDAV4	02A9 R0NDV6	02D7 RON1FC	01BF RSV	01D0 RSV1
01E2 RSV2	01ED RSV24	200B SAREA	2007 SC1BF	2008 SC0BF
2006 SD1BF	2009 S0DBF	01F3 SEDGE	0184 SEND	9800 S10C
9000 S10D	200A S0DBF	022B SOUT	204B STACK	2000 TSTM



§ 3-4

HP9816 - PC9801 間を GPIB を用いてファイル転送する

亘 慎一

現在、パソコンは異機種間の互換性がほとんどなく、ディスク・サイズも3.5インチ、5インチ、8インチとまちまちです。しかし、プログラムやデータ、あるいは、ディスク装置を共用したいことがしばしばあります。

このようなとき、RS-232CやGPIBを使ったパソコン間の通信が便利です。RS-232Cを使ったものについては比較的多く紹介されているので、GPIBによるデータ転送について紹介します(図1)。

接続例として、HP製のHP9816と日本電気製のPC9801を取り上げることにします。

ハードの設定

GPIBは、本来、計測器の制御用バスなので、パソコン同士をつなぐ場合、問題点がないわけではありません。GPIBでは、接続された機器の中でシステム・コントローラは1台でなければなりません。しかし、一般的に、パソコンは、コントローラになっているので、パソコン同士をつなぐ場合、GPIB上に複数のコ

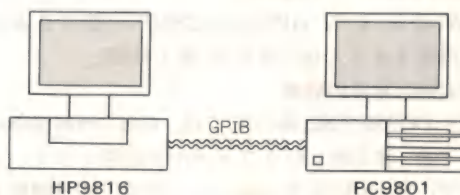
ントローラが存在することになってしまいます。このような場合、GPIB基板上的DIP-SWを使って、一方をスレーブに指定しなければなりません。

ここでは、HP9816をシステム・コントローラ、PC9801をスレーブ・モードとして接続することになります。

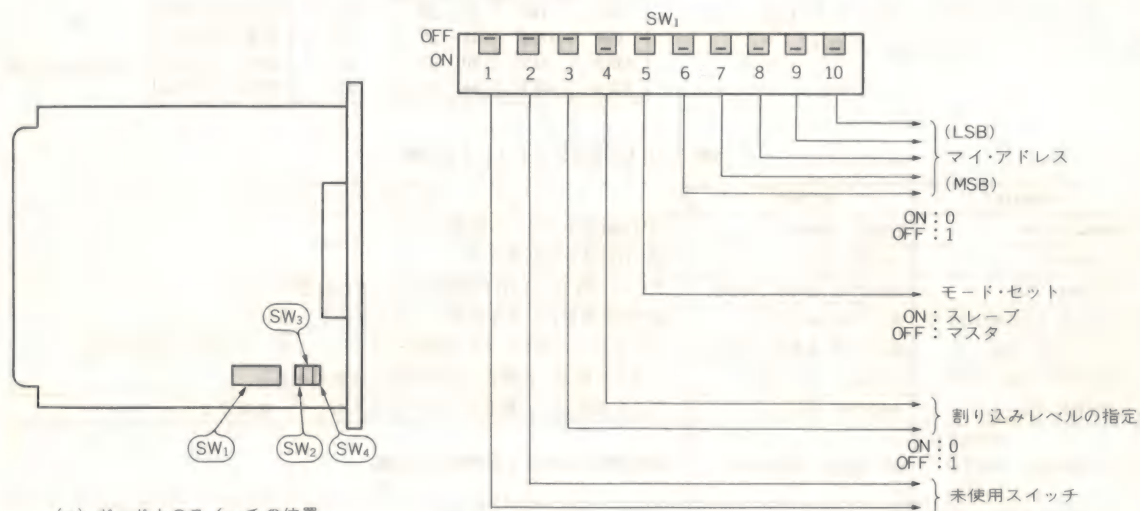
PC9801のGPIBカードの設定方法

GPIBカードを使うには、カード上のディップ・スイッチ、ジャンパ・スイッチ(図2)とPC9801本体のメモリ・スイッチをプリセットする必要があります。

〈図1〉 GPIBによる接続



〈図2〉⁽⁹⁾ PC9801のGPIBカードの設定



① マイ・アドレスの設定

SW₁の6～10でPC9801のアドレスを設定します。

② モード・セット

PC9801の GPIBカードには、マスタ・モードとスレーブ・モードと呼ばれる二つのモードがあります。

マスタ・モードというのは、PC9801がシステム・コントローラとなるモードで、各機器をトーカーやリスナに指定したりマルチ・ライン・メッセージを送出することができます。

スレーブ・モードというのは、GPIB上のコントローラからの指定により、トーカーやリスナになるモードで、PC9801はコントローラにはなりません。

これらのモードは、システムを起動させたときに、セットされていたSW₁₋₅の状態によって決まるので、それ以降はプログラムでモードを変更することはできません。

PC9801を使って測定器をコントロールする場合は、マスタ・モード(SW₁₋₅をOFF)にします。一方、GPIB上にほかのコントローラが存在し、その下でPC9801を働かせるときは、スレーブ・モード(SW₁₋₅をON)にします。HP9816とのファイル転送では、HP9816がコントローラになっているので、PC9801をスレーブ・モードにしています。

③ 割り込みレベルの設定

SW₁の3、4で、GPIBからCPUへの割り込みレベルを指定することができます(表1参照)。

④ PC9801本体の設定

MS-DOS版のN₈₈BASICでは、GPIB制御機能は、GPIB.EXEというファイルで提供されています。そこで、N₈₈BASICを起動するときに、拡張機能を使うことを宣言しなければなりません。

起動時に、

N₈₈BASIC/E:GPIB

とキーインしてやればよいのです。

(注)MS-DOS版でないN₈₈BASICの場合は、PC9801のメモリSW₄₋₅をONにする必要があります。

これで、GPIBカードの設定はOKです。

◆ PC9801のGPIBコマンドについて

HP9816にあるステートメントでPC9801にはないものがありますが、表2に示したようにして対応させることができます。

GPIB制御の測定器では、多くの場合、HP9816によるプログラミング例が付いていますが、この表を参考にすれば、HP9816用のプログラムをPC9801用に書き換えることができます。

◆ PC9801のGPIBプログラム例(スペアナをコントロール)

PC9801を使ったGPIBによる測定器のコントロールと、データ読み取りについて、HP社のスペクトル・アナライザHP8590Aを取り上げて説明します(図3)。

「スペアナをプリセットした後でシングル・スイープ・モードに設定する。次にスペアナの中心周波数を300MHzにセットし、1回スイープを行わせる。そして、スペアナの中心周波数を読み取り、画面に表示する。」

これを行わせるためのHP9816とPC9801のプログラムが、リスト1とリスト2です。HP9816では、CLEARによってプリセットしますが、PC9801では、ISET IFCを使います。

〈表1〉⁽¹⁹⁾ 割り込みレベルの設定

SW ₁		アドレス	ベクタ番号	用途
3	4			
ON	ON	2C~2F	B	拡張バスINT ₀
ON	OFF	48~4B	12	拡張バスINT ₄
OFF	ON	50~53	14	拡張バスINT ₅
OFF	OFF	54~57	15	拡張バスINT ₆

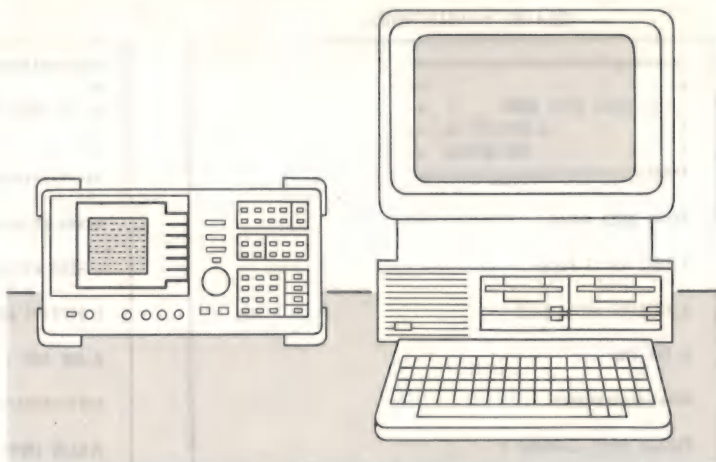
(出荷時設定値)

〈表2〉⁽²⁴⁾ GPIBステートメントの比較

HP9816	PC9801	ステートメントについて
REMOTE 7	ISET REN	GPIB機器をリモート状態にする
CLEAR 7	ISET IFC	GPIB機器を初期化する
LOCAL 701	WBYTE &H2L,&H1;	アドレス指定したGPIB機器をローカル状態にする
LOCAL LOCKOUT 7	WBYTE &H11;	GPIB機器をローカル制御にもどらないようにする
TRIGGER 701	WBYTE &H2L,&H8;	リスナとしてアドレスした機器トリガ・メッセージ(GET)を送信する
OUTPUT 701:"F1"	PRINT @1:"F1"	アドレス指定した機器に文字列または数値を送る
ENTER 701:A	INPUT @1:A	アドレス指定した機器からデータを入力して、変数にセット
GOSUB ON INTR 7 GOTO CALL	ON SRQ GOSUB	SRQ 割り込みによる分岐先の定義
ENABLE INTR 7;2	SRQ ON	GPIBのSRQ 割り込みの許可
SPOLL(701)	POLL 1,S	ステータス・レジスタの値を読みとる

〈図3〉

GPIBでスペアナHP8590Aをコントロール



次に、スペアナにコマンド(IP:初期化, SNGLS:シングル・スイープ・モード, CF 300MHz:中心周波数を300MHzに設定, TS:スイープ, CF?:中心周波数を出力)を送ります。最後に、スペアナから中心周波数を読み出します。

転送プログラム

HP9816とPC9801をつないで、アスキー・セーブされたプログラムのファイルの転送と、データ・ファイルの転送を行ってみることにします。このとき、HP9816はシステム・コントローラ、PC9801は、スレーブ・モードに設定します。

● アスキー・セーブされたファイルの転送

転送プログラム(章末に掲載リスト3, リスト4)を実行すると、画面が図4のようになり送信か受信かを聞いてくるので、1台を送信に、ほかの1台を受信に設定します。次に、転送するファイル名を入力すると

転送を開始します。

アスキー・セーブのしかたですが、HP9816では、SAVE“ファイル名”とすると、プログラムをアスキー・セーブすることができます。また、PC9801では、SAVE“ファイル名”, Aとすることによってアスキー・セーブすることができます。

MS-DOS版のN₈₈BASICでは、LOAD命令を実行したとき“.BAS”という拡張子が自動的に付加されてファイル・アクセスが行われます。そこで、HP9816からPC9801にファイルを転送する場合、PC9801側のファイル名に“.BAS”という拡張子を付加しておくといよいでしょう。

● データ・ファイルの転送

例として、図5のようにセーブされたデータ・ファイルの転送を行ってみます。

先ほどと同じように 送信, 受信の別とファイル名を入力することにより、転送を行うことができます(図6)。セーブされたデータの様式によって、プログラ

〈リスト1〉
HP9816による
プログラム

```
10 SPANA=718          ' スペアナのGP I B上のアドレス 18
20 CLEAR SPANA        ' スペアナの初期化
30 OUTPUT SPANA;"IP;SNGLS;" ' 初期化、シングルスイープモード
40 OUTPUT SPANA;"CF 300MHz;TS;" ' 中心周波数を300MHzに設定、スイープ
50 OUTPUT SPANA;"CF?;" ' 中心周波数出力
60 ENTER SPANA;CFRQ    ' 中心周波数の読み出し
70 PRINT "CENTER FREQUENCY =";CFRQ;"Hz" ' 読み出した周波数を画面に出力
80 END
```

〈リスト2〉
PC9801による
プログラム

```
10 SPANA=18          ' スペアナのGP I B上のアドレス 18
20 ISET IFC          ' スペアナの初期化
30 PRINT @SPANA;"IP;SNGLS;" ' 初期化、シングルスイープモード
40 PRINT @SPANA;"CF 300MHz;TS;" ' 中心周波数を300MHzに設定、スイープ
50 PRINT @SPANA;"CF?;" ' 中心周波数出力
60 INPUT @SPANA;CFRQ ' 中心周波数の読み出し
70 PRINT "CENTER FREQUENCY =";CFRQ;"Hz" ' 読み出した周波数を画面に出力
80 END
```

〈図4(a)〉 HP9816の画面

```

*****
*                                     *
* << ASCII FILE SEND               *
*                                     *
*           & RECEIVE >>          *
*                                     *
*           FOR HP9816              *
*                                     *
*****

***** MENU *****

1.SEND ASCII FILE

2.RECEIVE ASCII FILE

3.JOB END

*****

PLEASE INPUT COMMAND =

```

〈図4(b)〉 PC9801の画面

```

*****
*                                     *
* << ASCII FILE SEND               *
*                                     *
*           & RECEIVE >>          *
*                                     *
*           FOR PC-9801            *
*                                     *
*****

***** MENU *****

1.SEND ASCII FILE

2.RECEIVE ASCII FILE

3.JOB END

*****

PLEASE INPUT COMMAND =

```

ム(章末に掲載リスト5, リスト6)を適当に変更してやれば, 必要なデータの転送が可能となります。

(注)MS-DOS上のN₈₈BASICを使っているので, PC9801のディスク・ドライブの指定は, A:, B:, ...としています。

● おわりに

ここでは, GPIBによるデータ転送の基礎的なプログラムを紹介しました。ここで紹介したPC9801用のプログラムはN₈₈BASICで書いてあるので, PC9801同士あるいはPC9801とPC8801のファイル転送やデータ転送にも使うことができます。

この場合も一方をマスタ・モード, 他方をスレーブ・モードにしてやる必要があります。また, MS-DOS版のN₈₈BASICかどうかにより, ディスク・ドライブの指定の仕方を変更する必要があります。

入力したファイルがなかったときやディスク一杯になってしまったときなどのエラー処理は, ここで紹介したプログラムでは行っていませんが, 必要なら自分でトライしてみてください。

また, 転送するデータのフォーマットが違う場合は, それに合わせてプログラムを変更してください。

● 参考・引用文献

- (1)*松井雅行, 竹尾佳己: GPIBコントローラ, トランジスタ技術, 1983年1月号。
- (2)*松崎幹夫: GPIBの使い方, トランジスタ技術, 1983年12月号。
- (3)*松井雅行, 竹尾佳己: GPIBインターフェース成功への手法, トランジスタ技術, 1984年1月号。
- (4)*日本電気, μ PD7210アプリケーション・ノート, 1982年4月号。
- (5)*川村道生, 檜山哲夫: GPIBインターフェースの製作, トランジスタ技術, 1982年1月号。
- (6)*Motorola, MC68488データ・シート, 1980年。

〈図5〉セーブされるデータ・フォーマット

HEAD(1)	HEAD(2)	HEAD(22)
DATA(1,1)	DATA(1,2)	DATA(1,12)
DATA(2,1)	DATA(2,2)	DATA(2,12)
		
		
		
		
DATA(N,1)	DATA(N,2)	DATA(N,12)

- (7)*TI社, The Bipolar Digital, Integrated Circuits Data Book, 1985年。
- (8) 鬼頭史郎: 標準インターフェース・バス (HP-IB), インターフェース, 1977年, 2月号, p.41, CQ出版社。
- (9) 岡村勉夫: IEEE-488標準デジタル・バスとその応用, インターフェース, 1979年, 2月号, p.45, CQ出版社。
- (10) 岡村勉夫: IEEE-488標準デジタル・バスとその応用, インターフェース, 1980年, 7月号, p.70, CQ出版社。
- (11) 神谷峰夫: GP-IBインターフェースの設計と問題点, インターフェース, 1980年, 8月号, p.70, CQ出版社。
- (12) 岡村勉夫: 標準デジタル・バス (IEEE-488) とそのソフトウェア, インターフェース, 1983年, 11月号, p.189, CQ出版社。
- (13) 佐倉成之: 光ファイバ信号伝送回路の製作と実験, トランジスタ技術, 1983年, 9月号, p.355。
- (14) インターシル, IM6402, インターシル・デジタル・プロダクツ, 1978年, pp.8~125。
- (15)*東芝, 東芝光伝送リンク, テクニカル・データ, 1982年。
- (16) マイクロサイエンス, UIO-488B, 取扱説明書。
- (17) マイクロサイエンス, SBD-488T, 取扱説明書。

〈図 6 (a)〉 HP9816の画面

```

*****
*                                     *
* << DATA FILE SEND                 *
*      & RECEIVE >>                  *
*      FOR HP9816                     *
*                                     *
*****

***** MENU *****

1.SEND DATA

2.RECEIVE DATA

3.JOB END

*****

PLEASE INPUT COMMAND =

```

〈図 6 (b)〉 PC9801の画面

```

*****
*                                     *
* << DATA FILE SEND                 *
*      & RECEIVE >>                  *
*      FOR PC-9801                   *
*                                     *
*****

***** MENU *****

1.SEND DATA

2.RECEIVE DATA

3.JOB END

*****

PLEASE INPUT COMMAND =

```

(18) シャープ、MZ80B, オーナーズマニュアル/GPIBインターフェース。

(19)*日本電気、GP-IB (IEEE-488) インターフェースボードユーザーズマニュアル, pp.17~19.

(20) YHP, モデル16 BASIC操作入門。

(21) YHP, BASIC I/Oプログラミング。

(22) HP, 8590A, スペクトラム・アナライザオペレーティング・マニュアル。

(23) HP, 8590A Portalbe RF Spectrum Analyzer Programming Manual HP-IB.

(24)*インターフェース別冊 GPIBプログラミング・ノート, インターフェース, 1987年, 4月号, CQ出版社。

好評発売中

計測制御の信号処理からセンサ/通信インターフェースまで

モジュール化に役立つ実用電子回路集

トランジスタ技術編集部編 B5判 160頁 定価1,631円(税込)

どんなにアイデアやオリジナリティに溢れた回路でも、長大で複雑だったり部品の入手が困難では、応用範囲が狭くなってしまいます。本書では、あらゆる場面で役立つ、モジュール設計のための回路として、汎用部品でコンパクトに構成した粋な回路を集めました。

また設計した回路を、より実用的なものにするには、いろいろなインターフェース技術が要求されます。モジュール化設計した回路同士やパソコン、測定器との接続などに役立つ、便利なインターフェース回路も豊富に紹介しています。

第1章 必要な信号へ変換する計測用信号処理回路
第2章 必要な信号・電圧をつくる発振&電源回路
第3章 必要な信号を取り出し増幅するアンプ&フィルタ回路
第4章 高い精度と耐ノイズ性を確保するセンサ&計測インターフェース

第5章 音と映像の信号を自在に操るオーディオ&ビデオ信号処理回路
第6章 ACコントロールとメカトロニクスに役立つパワー・エレクトロニクス関連汎用インターフェースを幅広く活用する
第7章 通信インターフェース回路

ハードウェア・デザイン・シリーズ②



```

1000 !*****
1010 !*
1020 !* << SEND & RECEIVE ASCII FILE >> *
1030 !* FOR HP9816 *
1040 !*****
1050 !
1060 !***** INIT *****
1070 !
1080 DIM Source_files[6],Desti_file[6]
1090 DIM D[255]
1100 !
1110 !/*****
1120 !
1130 Pc_addr=711 !PC-9801 HP-IB ADDRESS
1140 CLEAR Pc_addr !CLEAR HP-IB
1150 !
1160 PRINT "*****
1170 PRINT "*"
1180 PRINT "*" <<ASCII FILE SEND
1190 PRINT "*" & RECEIVE>>
1200 PRINT "*" FOR HP9816
1210 PRINT "*****
1220 PRINT
1230 !
1240 Menu: PRINT "***** MENU *****"
1250 PRINT
1260 PRINT "1.SEND ASCII FILE"
1270 PRINT
1280 PRINT "2.RECEIVE ASCII FILE"
1290 PRINT
1300 PRINT "3.JOB END"
1310 PRINT
1320 PRINT "*****
1330 PRINT
1340 !
1350 INPUT "PLEASE INPUT COMMAND = ",N
1360 IF N<1 OR N>3 THEN Menu
1370 IF N=1 THEN Fsend
1380 IF N=2 THEN Frec
1390 STOP
1400 !

```

```

1410 Fsend:
1420 PRINT "*"
1430 PRINT "*" SEND ASCII FILE *
1440 PRINT "*" HP9816 =====>PC-9801 *
1450 PRINT "*"
1460 PRINT "*****
1470 PRINT
1480 !
1490 INPUT "SOURCE_FILE NAME (HP9816)=" ,Source_fname$
1500 !
1510 INPUT "DESTINATION_FILE NAME (PC-9801)=" ,Desti_fname$
1520 IF Desti_fname$="" THEN Desti_fname$=Source_fname$
1530 !
1540 PRINT Source_fname$,(HP9816) =====> ",Desti_fname$,(PC-
9801)"
1550 PRINT
1560 !
1570 INPUT " O.K. =====>Y (Yes) ENTER OR N (No) ENTER",K$
1580 IF K$="Y" THEN St_send
1590 GOTO Fsend
1600 !
1610 St_send: PRINT "..... NOW BEGIN SEND ASCII FILE ....."
1620 PRINT
1630 !
1640 !***** OPEN_FILE *****
1650 !
1660 ASSIGN @Read_disk TO Source_fname$,"HP82901,700,1"
1670 ON END @Read_disk GOTO F_end
1680 !
1690 OUTPUT Pc_addr;Desti_fname$
1700 Send_loop:ENTER @Read_disk;D$
1710 OUTPUT Pc_addr;D$
1720 GOTO Send_loop
1730 !
1740 F_end: OUTPUT Pc_addr;"EOF"
1750 ASSIGN @Read_disk TO *
1760 PRINT "..... SEND ASCII FILE END ....."
1770 PRINT
1780 GOTO Menu
1790 !
1800 Frec: PRINT "*****

```


〈リスト4〉 PC9801用のアスキー・ファイル転送プログラム(つづき)

```

1420 '
1430 *FSEND
1440 ,
1450 PRINT"*****"
1460 PRINT" *
1470 PRINT" * SEND-ASCII FILE *
1480 PRINT" * PC-9801 ==> HP9816 *
1490 PRINT" *
1500 PRINT"*****"
1510 PRINT
1520 ,
1530 INPUT"INSERT FILE TO #2 (PC-9801) O.K. ==> Push ENTER";K$
1540 PRINT
1550 ,
1560 LINE INPUT;PNAME$
1570 PRINT" FILE NAME : ";PNAME$
1580 PRINT
1590 ,
1600 ,***** OPEN FILE *****
1610 ,
1620 OPEN "B:~PNAME$ FOR INPUT AS #1
1630 ,
1640 ,*****
1650 ,
1660 PRINT "... NOW SENDING FILE ..."
1670 PRINT
1680 GOSUB *SENDF
1690 ,
1700 PRINT "**** FILE SEND COMPLETED ****"
1710 PRINT
1720 ,
1730 FOR I=1 TO 1000 :NEXT I
1740 ,
1750 CLS 3
1760 RETURN
1770 ,
1780 *SENDF
1790 IF EOF(1) THEN *SENDEND
1800 LINE INPUT #1,D$
1810 PRINT:D$
1820 GOTO *SENDF
1830 ,

```

: 'ファイルより入力
: 'GP-IBへの出力

: 'GP-IBよりファイル名入力

```

1840 *SENDEND
1850 CLOSE
1860 PRINT:;EOF"
1870 RETURN
1880 ,
1890 ,*****
1900 , *
1910 , * RECEIVE-ASCII FILE *
1920 , * HP9816 ==> PC-9801 *
1930 , *
1940 ,*****
1950 ,
1960 *FREC
1970 ,
1980 PRINT"*****"
1990 PRINT" *
2000 PRINT" * RECEIVE-ASCII FILE *
2010 PRINT" * HP9816 ==> PC-9801 *
2020 PRINT" *
2030 PRINT"*****"
2040 PRINT
2050 ,
2060 INPUT"INSERT FILE TO #2 (PC-9801) O.K. ==> Push ENTER";K$
2070 PRINT
2080 ,
2090 LINE INPUT;PNAME$
2100 PRINT" FILE NAME : ";PNAME$
2110 PRINT
2120 ,
2130 ,***** OPEN FILE *****
2140 ,
2150 OPEN "B:~PNAME$ FOR OUTPUT AS #1
2160 ,
2170 ,*****
2180 ,
2190 PRINT "... NOW RECEIVING FILE ..."
2200 PRINT
2210 GOSUB *RECIF
2220 ,
2230 PRINT "**** FILE RECEIVE COMPLETED ****"
2240 PRINT
2250 ,

```

: 'GP-IBよりファイル名入力

＜リスト4＞ PC9801用のアスキー・ファイル転送プログラム(つづき)

```

2260 FOR I=1 TO 1000 :NEXT I
2270   :
2280 CLS 3
2290 RETURN
2300   :
2310 *RECF
2320 LINE INPUT D$:
2330 IF D$="EOF" THEN *RECFD
2340 PRINT #1, D$
2350 GOTO *RECF
2360   :
2370 *RECFD
2380 CLOSE
2390 RETURN

```

:' GP - I B より入力
:' ファイルへの出力

＜リスト5＞ HP9816用のデータ転送プログラム(つづき)

```

1160 !
1170   PC_addr=711      !PC-9801 HP-IB ADDRESS
1180   CLEAR PC_addr    !CLEAR HP-IB
1190   !
1200   PRINT "*****"
1210   PRINT "x"
1220   PRINT "x" <<DATA FILE SEND
1230   PRINT "x" & RECEIVE>
1240   PRINT "x" FOR HP9816
1250   PRINT "*****"
1260   PRINT
1270   !
1280   Menu:
1290   PRINT "***** MENU *****"
1300   PRINT
1310   PRINT "1.SEND DATA FILE"
1320   PRINT "2.RECEIVE DATA FILE"
1330   PRINT
1340   PRINT "3.JOB END"
1350   PRINT
1360   PRINT "*****"
1370   PRINT
1380   !
1390   INPUT "PLEASE INPUT COMMAND = ", N
1400   IF N<1 OR N>3 THEN Menu
1410   IF N=1 THEN Fsend
1420   IF N=2 THEN Frec
1430   STOP
1440   !
1450   Fsend:
1460   PRINT "*****"
1470   PRINT "x" SEND DATA FILE
1480   PRINT "x" HP9816 ==>PC-9801
1490   PRINT "x"
1500   PRINT "*****"
1510   PRINT
1520   !
1530   INPUT "SOURCE_FILE NAME (HP9816)=", Source_fname$
1540   !
1550   INPUT "DESTINATION_FILE NAME (PC-9801)=", Desti_fname$
1560   IF Desti_fname$="" THEN Desti_fname$=Source_fname$
1570   !

```

＜リスト5＞ HP9816用のデータ転送プログラム

```

1000 !*****
1010 !*
1020 !* << SEND & RECEIVE DATA FILE >>
1030 !* FOR HP9816
1040 !*****
1050 !
1060 !***** INT. *****
1070 !
1080   OPTION BASE 1
1090   INTEGER Uti(6), Scan1, Scant
1100   INTEGER Head(22), D(12)
1110   INTEGER Nsize
1120   DIM Source_files$(8), Desti_files$(8)
1130   DIM Dn$(10), Hd$(22)[10], Dat$(1019, 12)[10]
1140   !
1150   !

```

```

1580 PRINT Source_fname$;"(HP9816) ===== ";Desti_fname$;"(PC-
9801)"
1590 PRINT
1600 !
1610 INPUT " O.K. =====>Y (Yes) ENTER OR N (No) ENTER",K$
1620 IF K$="Y" THEN St_send
1630 GOTO Fsend
1640 !
1650 St_send: PRINT "..... NOW BEGIN SEND DATA FILE ....."
1660 PRINT
1670 !
1680 !*****x OPEN_FILE *****x
1690 !
1700 ASSIGN @Read_disk TO Source_fname$;"HP82901,700,1"
1710 ON END @Read_disk GOTO No_data
1720 ENTER @Read_disk;Head(*)
1730 !
1740 FOR I=1 TO 6
1750   Uti(I)=Head(I)
1760 NEXT I
1770 Scanl=Head(7)
1780 Scant=Head(8)
1790 Scanw=Head(9)*.001
1800 PRINT USING @maghead;Uti(*),Scanl,Scant,Scanw
1810 Nsize=Scanl*Scant-I
1820 PRINT "Nsize=";Nsize
1830 OUTPUT Pc_addr;Desti_fname$
1840 OUTPUT Pc_addr;Nsize
1850 FOR I=1 TO 22
1860   OUTPUT Pc_addr;Head(I)
1870 NEXT I
1880 FOR I=1 TO Nsize
1890   DISP I
1900   IF (I MOD 10)=0 THEN
1910     PRINT " ";
1920     IF (I MOD 500)=0 THEN PRINT
1930   END IF
1940   ENTER @Read_disk;D(*)
1950   FOR J=1 TO 12
1960     OUTPUT Pc_addr;D(J)
1970   NEXT J
1980 NEXT I

```

```

1990
2000 No_data:
2010 PRINT
2020 PRINT "..... SEND DATA FILE END ....."
2030 PRINT
2040 GOTO Menu
2050 !
2060 Frec:
2070 PRINT "*****x *****x"
2080 PRINT "x"
2090 PRINT "x" RECEIVE DATA FILE "x"
2100 PRINT "x" PC-9801 =====>HP9816 "x"
2110 PRINT "x"
2120 PRINT "*****x *****x"
2130 PRINT
2140 !
2150 INPUT "SOURCE_FILE NAME (PC-9801)=",Source_fname$
2160 !
2170 INPUT "DESTINATION_FILE NAME (HP9816)=",Desti_fname$
2180 IF Desti_fname$="" THEN Desti_fname$=Source_fname$
2190 !
2200 PRINT Source_fname$;"(PC-9801) ===== ";Desti_fname$;"(HP
9816)"
2210 PRINT
2220 INPUT " O.K. =====>Y (Yes) ENTER OR N (No) ENTER",K$
2230 IF K$="Y" THEN St_rec
2240 GOTO Frec
2250 !
2260 St_rec: PRINT "..... NOW BEGIN RECEIVE DATA FILE ....."
2270 PRINT
2280 !
2290 !*****x OPEN_FILE *****x
2300 !
2310 CREATE BDAT Desti_fname$;"HP82901,700,1",144
2320 ASSIGN @Write_disk TO Desti_fname$;"HP82901,700,1"
2330 !
2340 OUTPUT Pc_addr;Source_fname$
2350 !
2360 ENTER Pc_addr;Dn$
2370 Nsize=VAL(Dn$)
2380 PRINT "Nsize=";Nsize

```


くリスト6> PC9801用のデータ転送プログラム(つづき)

```

1410 'x PC-9801 ==> HP9816 *
1420 'x
1430 '*****
1440 '
1450 *DSEND
1460 '
1470 PRINT"*****"
1480 PRINT"x"
1490 PRINT"x" SEND-DATA
1500 PRINT"x PC-9801 ==> HP9816 *
1510 PRINT"x"
1520 PRINT"*****"
1530 PRINT
1540 '
1550 INPUT"INSERT DATA FILE TO #2 (PC-9801) 0.K. ==> Push ENTER":K$
1560 PRINT
1570 '
1580 LINE INPUT:DNAME$
1590 PRINT"SEND DATA FILE NAME : ";DNAME$
1600 PRINT
1610 '
1620 '***** OPEN FILE *****
1630 '
1640 OPEN "B:"-DNAME$ FOR INPUT AS #1
1650 '
1660 '*****
1670 '
1680 FOR I=1 TO 22
1690 INPUT #1,HEAD(1)
1700 NEXT I
1710 '
1720 PRINT HEAD(1);"/";HEAD(2);"/";HEAD(3);" ";
1730 PRINT HEAD(4);"/";HEAD(5);"/";HEAD(6);
1740 PRINT " L=";HEAD(7);" T=";HEAD(8);" W=";HEAD(9)*.001
1750 PRINT
1760 '
1770 DN=HEAD(7)*HEAD(8)-1
1780 '
1790 NSIZE$=STR$(DN)

```

: 'GP-I Bよりファイル名入力

```

1800 PRINT:NSIZE$
1810 PRINT"Size=";NSIZE$
1820 PRINT
1830 '
1840 FOR I=1 TO 22
1850 HD$(I)=STR$(HEAD(I))
1860 PRINT:HD$(I)
1870 NEXT I
1880 '
1890 FOR I=1 TO DN
1900 IF (I MOD 10)=0 THEN PRINT"*";
1910 IF (I MOD 500)=0 THEN PRINT
1920 FOR J=1 TO 12
1930 INPUT #1,DD(J)
1940 NEXT J
1950 FOR J=1 TO 12
1960 DAT$(J)=STR$(DD(J))
1970 PRINT:DAT$(J)
1980 NEXT J
1990 NEXT I
2000 '
2010 '***** CLOSE FILE *****
2020 '
2030 CLOSE
2040 '
2050 '*****
2060 '
2070 PRINT
2080 PRINT"DATA SEND END"
2090 PRINT
2100 RETURN
2110 '
2120 '*****
2130 'x
2140 'x RECEIVE-DATA
2150 'x HP9816 ==> PC-9801
2160 'x
2170 '*****
2180 '

```

: 'GP-I Bへヘッド出力

: 'ファイルよりデータ入力

: 'GP-I Bへデータ出力

: 'GP-I Bへデータ数出力


```

2190 *DREC
2200 ,
2210 PRINT"*****"
2220 PRINT" "
2230 PRINT" "
2240 PRINT" "
2250 PRINT" "
2260 PRINT" "
2270 PRINT
2280 ,
2290 INPUT"INSERT DATA FILE TO #2 (PC-9801) O.K. ==> Push ENTER";K$
2300 PRINT
2310 ,
2320 LINE INPUT:DNAME$
2330 PRINT"RECEIVE DATA FILE NAME : ";DNAME$
2340 PRINT
2350 ,
2360 INPUT:DN
2370 PRINT"Size=";DN
2380 PRINT
2390 ,
2400 FOR I=1 TO 22
2410 INPUT:HEAD(I)
2420 NEXT I
2430 ,
2440 PRINT HEAD(1);"/";HEAD(2);"/";HEAD(3);" ";
2450 PRINT HEAD(4);"/";HEAD(5);"/";HEAD(6);
2460 PRINT " L=";HEAD(7);" T=";HEAD(8);" W=";HEAD(9)*.001
2470 PRINT
2480 ,
2490 ,***** OPEN FILE *****

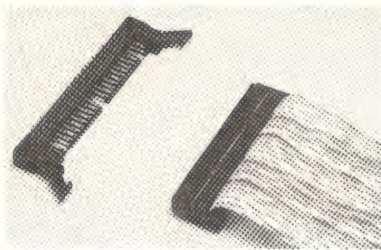
2500 ,
2510 OPEN "B:"+DNAME$ FOR OUTPUT AS #1
2520 ,
2530 ,*****
2540 ,
2550 FOR I=1 TO 22
2560 WRITE #1,HEAD(I)
2570 NEXT I
2580 ,
2590 FOR I=1 TO DN
2600 IF (I MOD 10)=0 THEN PRINT"*";
2610 IF (I MOD 500)=0 THEN PRINT
2620 FOR J=1 TO 12
2630 INPUT:DD(J)
2640 NEXT J
2650 FOR J=1 TO 12
2660 WRITE #1,DD(J)
2670 NEXT J
2680 NEXT I
2690 ,
2700 ,***** CLOSE FILE *****
2710 ,
2720 CLOSE
2730 ,
2740 ,*****
2750 ,
2760 PRINT
2770 PRINT"DATA RECEIVE END"
2780 PRINT
2790 RETURN

:'ファイルヘッダ出力

:'GP-I Bよりデータ入力

:'ファイルヘッダ出力

```



§ 4-1

SCSI インターフェースの基礎

里 和政

SCSI(Small Computer System Interface)は、この意味のとおりマイクロコンピュータ・システム向けに考案されたインターフェースです。標準化も進んでおり、1986年7月にANSI(米国規格協会)において規格化されました。

SCSIの基礎は、ハード・ディスク装置用のインターフェースとして使用されていた、SASI(Shugart Associates System Interface)を拡張しています。

SASIは、複雑な周辺装置の制御をホストで行うのではなくコントローラに任せ、簡単なインターフェースによって制御する方法です。

最近では、SCSIを標準とした周辺装置が多くあり、ハード・ディスク装置、光ディスク装置、CD-ROM、プリンタ、計測器などが代表的なものです。

● SCSIバスの構成

SCSIの長所は、SCSIバス上に多くの装置を接続して、同一のプロトコルで制御でき、高速で大量のデータ転送も行えます。同一バス上には、最大8台まで接続することが可能です。

SCSIバス上の装置には、すべてIDと呼ばれる0～7(データ・バスに対応)の装置番号をもっており、この番号によって、装置の識別を行います。

実際の例を図1に示します。この例は、もっとも簡

単な構成です。

ホスト側のSCSIの制御を行うコントローラなどをホスト・アダプタと呼びます。

図2は、SCSIの特徴を出した構成です。複数のホスト・コンピュータ、複数の周辺装置を接続することができ、SASIと大きく異なります。

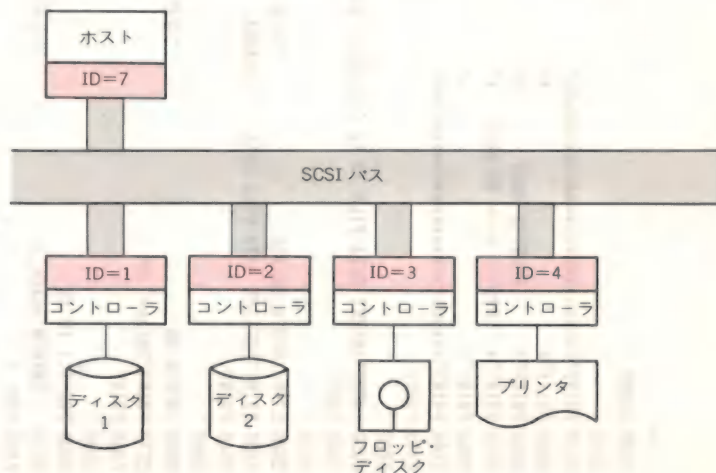
SCSIバス上では、ホストなどの命令を出す側をイニシエータと呼び、ディスク装置などの命令を受け取る側をターゲットと呼びます。しかし、イニシエータであれ、ターゲットであれ装置にはかわりはありません。

ホストは、ある時イニシエータであり、またある時ターゲットである場合が存在します。

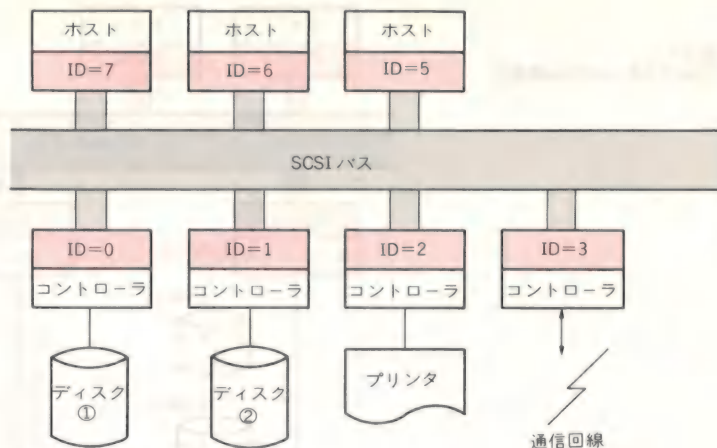
装置には、論理ユニット番号(LUN)が付けられており、最大7台までのユニットを接続することができます。LUNを使うことにより図3のような構成も可能です。

SCSIの特徴は、マルチタスク・システムでその能力を発揮することです。なぜならば、以前までのインターフェースでは、ひとつの装置が処理中であれば、その装置がバスを占有して、ほかの装置の処理が行えませんでした。SCSIでは、必要に応じてバスを解放することができるからです。

〈図1〉
SCSIバスのシステム構成(1)



〈図2〉
SCSIバスのシステム構成(2)



SCSIバスの信号

SCSIバスは、9本のデータ線(うち1本は、パリティ

ィ線)と9本の制御線で構成されます。

● BSY信号：SCSIバスを使用中であることを、ほかのコントローラに示します。またバスを占有する場合に、使用権の調停を行うときに使います。

論理ユニット

論理ユニットとは、一つのSCSIデバイスがコントロールしている物理的またはバーチャルな装置のことをいいます。通常8台まで(ユニット0～ユニット7)が、SCSIコマンド中図A(a)または“アイデンティファイ”メッセージ中図(b)で、ベンダ・ユニークに指定されます。

普通の小規模システムでは、これでも十分おつりがくるほどですが、SCSI規格では、名前に反して(?)2048台までの論理ユニット番号が、オプションで指定できるようになっています。これは、拡張メッセージである“エクステンデッド・アイデンティファイ(Extended Identify)”メッセージで行います。

4バイト長の拡張メッセージで、第1バイトは拡張メッセージであることを示し(01H)、第2バイトはメッセージの長さ(トータル・バイト数-2=0

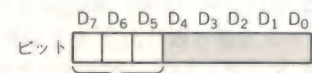
2H)、第3バイトが“エクステンデッド・アイデンティファイ”のコード(02H)、そして第4バイトで一つの論理ユニット中の〔これは(A)または(B)で指定される〕0～255までのサブ論理ユニット番号を指定します図(c)。これによって、一つのSCSIデバイスは、

$$\begin{array}{ccc} 8 & \times & 256 \\ \uparrow & & \uparrow \\ \text{論理ユニット} & & \text{サブ論理ユニット} \end{array} = 2048$$

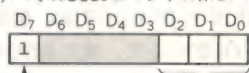
までの論理ユニットを扱うことができます。注意したいのは、SCSIバス上には、あくまで8台までのSCSIデバイス(イニシエータ+ターゲット)しか接続できないということです。

なお参考までに、 GPIB(IEEE-488)では同一バス上には最大15デバイスまで、また、1デバイスには、31台までの2次アドレスで指定できるユニットが接続できます。

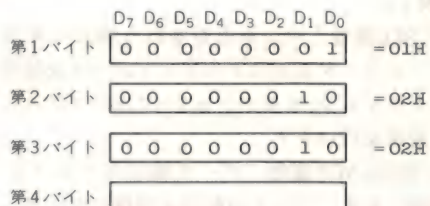
〈図A〉
論理ユニット
の指定



(a) すべてのSCSIコマンドの第2バイト

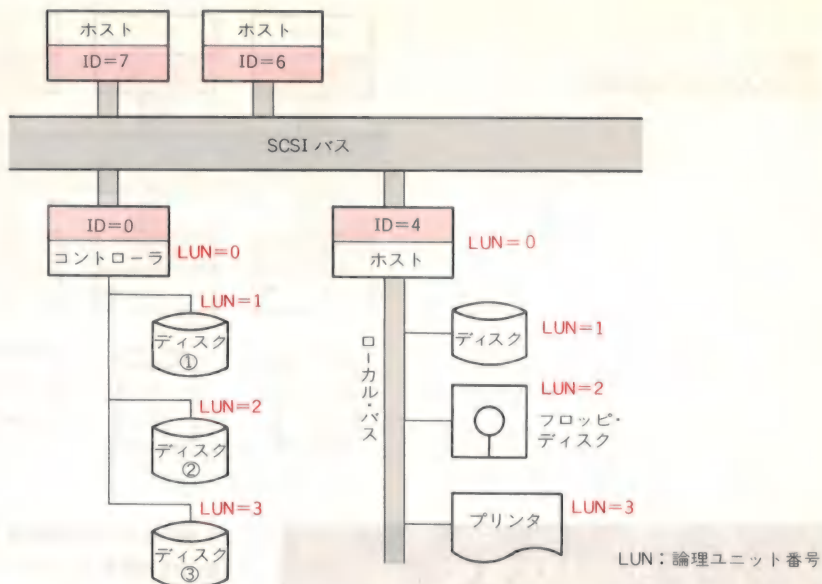


(b) “アイデンティファイ”メッセージ・バイト



(c) “エクステンデッド・アイデンティファイ”メッセージ

〈図3〉
SCSIバスのシステム構成(3)



〈表1〉
MSG, C/D, I/O信号とフェーズ

フェーズ名	MSG	C/D	I/O	転送方向	
				I	T
コマンド・フェーズ	0	1	0	→	
ステータス・フェーズ	0	1	1	←	
データ・イン・フェーズ	0	0	1	←	
データ・アウト・フェーズ	0	0	0	→	
メッセージ・イン・フェーズ	1	0	1	←	
メッセージ・アウト・フェーズ	1	0	0	→	

I: イニシエータ
T: ターゲット

SCSIバスは負理論であるため、“0”で“H”、“1”で“L”になる。

● **ATN信号**: 必要に応じて、いつでもイニシエータから、ターゲットに送ることができます。これは、ターゲットがフェーズの優先権をもっているため、イニシエータからの要求メッセージ(メッセージ・アウト)を送る場合使用します。

● **RST信号**: SCSIバスの解放をおこないます。この信号がアクティブとなるとすべての装置は、その状態にかかわらずバスの解放をしなければなりません。したがって、不用意にこの信号をアクティブにすると、ほかの装置の処理を中止させることになります。

● **MSG, C/D, I/O信号**: フェーズの状態を示します。これらの信号によって実行中のフェーズを示します(表1)。

● **SEL信号**: バスを占有を行う場合に使用します。イニシエータまたはターゲットがバスを使用するとき、セレクション、リセレクション・フェーズに入って、バスの確保を行います。

● **REQ, ACK信号**: データ転送のハンドシェイクを制御します。データの転送も同様にターゲット側が権利をもっています。

■ **データ・アウトの手順(イニシエータ・データ送信, 図4)**

- (1) ターゲットがREQをアクティブにする。
- (2) イニシエータは、データを出力して、ACKをアクティブにする。
- (3) ターゲットは、ACKがアクティブになるとデータを読み取り、REQをOFFにする。
- (4) イニシエータは、REQがOFFになるとACKをOFFにして、次のREQを待つ。

■ **データ・インの手順(イニシエータ・データ受信, 図5)**

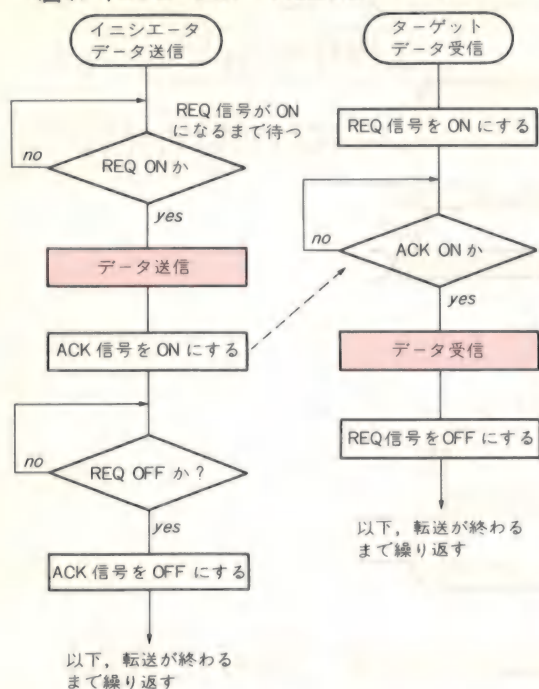
- (1) ターゲットがデータを出力してREQをアクティブにする。
- (2) イニシエータは、REQがアクティブになると、データを読み取りACKをアクティブにする。
- (3) ターゲットは、ACKがアクティブになると、REQをOFFにする。
- (4) イニシエータは、REQがOFFになるとACKをOFFにして、次のREQを待つ。

以上の手順にしたがってハンドシェイクを行います。

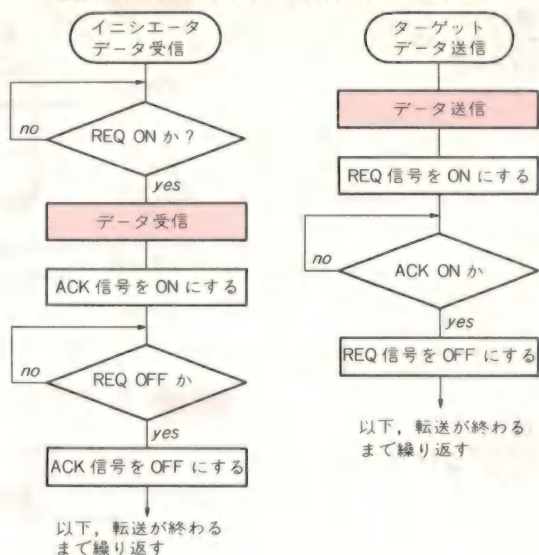
バスの各フェーズ

SCSIバスは、イニシエータとターゲット間をフェ

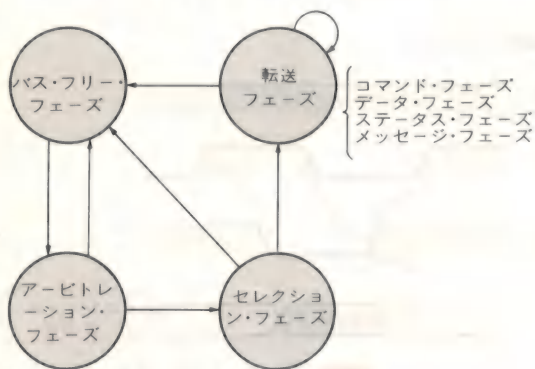
〈図4〉 イニシエータのデータ送信手順のフローチャート



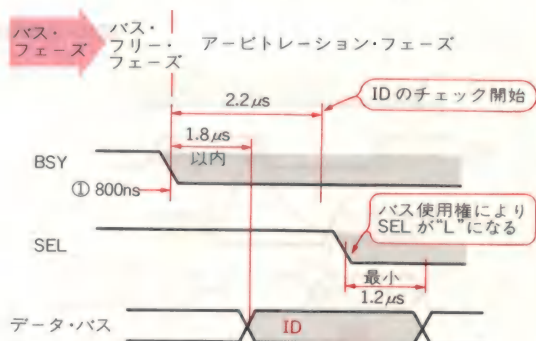
〈図5〉 イニシエータのデータ受信のフローチャート



〈図6〉 フェーズの移行



〈図7〉 アービトレーションのタイミング



① バス・フリー検出後、最小 800ns (バス・フリー・ディレイ)

ーズによって制御します。

図6にバス・フェーズを示します。

● バス・フリー・フェーズ

SCSIバスをだれも使用していない状態です。ハードウェア、ソフトウェア・リセット後には、本フェーズに入ります。この状態では、BSYがOFFになっています。

● アービトレーション・フェーズ

本フェーズによってバスの使用権を獲得します。このフェーズは、バス・フリーの状態から入ることができます。

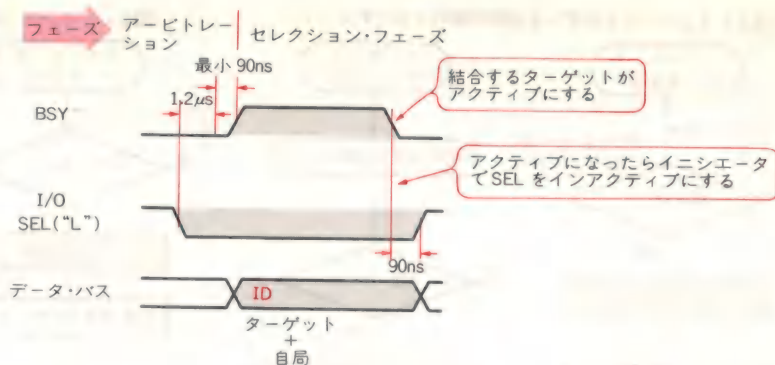
バスを使用したい装置は、BSYをアクティブにして自分の装置番号をデータ・バス上に出力します。こ

のときデータ・バス上のIDは、ビット0～ビット7のどれかに対応しています。

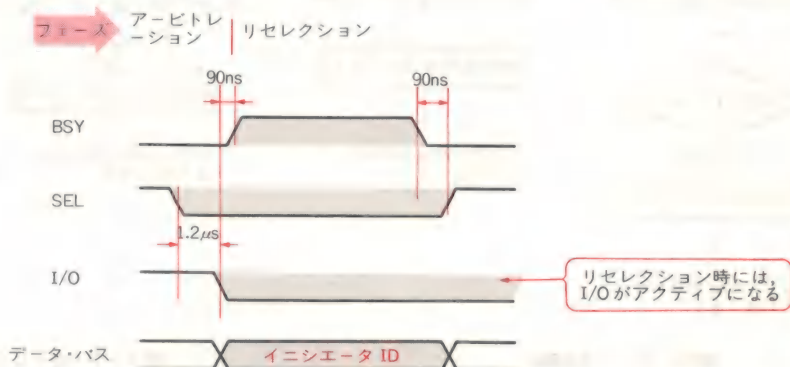
ほかの装置も同様にバスを使用したいならば、IDを出力します。最初にBSYがアクティブになってから2.2μs後に、アービトレーション(調停)が行われます。アービトレーションでは、自分よりも大きいIDが出力されているかを調べ、もし自分よりも大きいIDがなければ、バスの占有権を得てSELをアクティブにして、次のセレクション・フェーズまたはリセレクション・フェーズを開始します。

このフェーズは、複数のホスト(イニシエータ)があった場合に必要になり、単一のホストではとくに必要

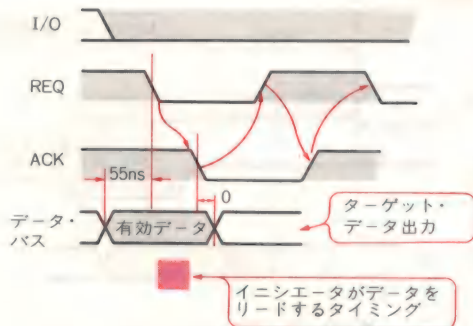
〈図8〉
セレクションの
タイミング



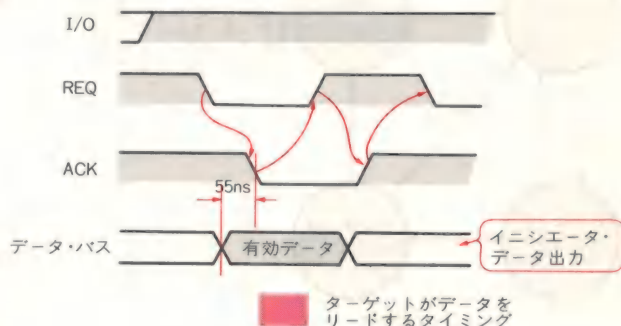
〈図9〉
リセレクションの
タイミング



〈図10〉 非同期データ転送のタイミング



(1) ターゲット→イニシエータ



(2) イニシエータ→ターゲット

なく、次のセレクション・フェーズを開始します(図7)。

● セレクション・フェーズ

アービトレーションによってバスを占有したイニシエータは、自分のIDとアクセスしたい装置のIDをバス上に出力し、BSYをOFFにします。それにより、アクセスされた装置は、結合可能であればBSYをアクティブにして結合します。その後、SELをOFFにします。

自分のIDは、リセレクション・フェーズがあった場合に使用されます(図8)。

以上で、バス上の接続を完了し、コマンド、データの転送を行います。

● リセレクション・フェーズ

セレクションと同様ですが、ターゲット自らバスの占有を行う点が異なります。この機能が、従来のSASIと大きく違う所です。前にも述べたように、デバイス上で時間のかかる処理を行うとバスが不要に占

〈図11〉 コマンド・グループCDBの型式

バイト \ ビット	7	6	5	4	3	2	1	0					
0	0	0	0	コマンド・コード									
1	LUN			論理ブロック・アドレス(MSB)									
2	論理ブロック・アドレス												
3	論理ブロック・アドレス (LSB)												
4	転送サイズ												
5	コントロール・バイト						フラグ	リンク					

(a) グループ0 CDB

バイト \ ビット	7	6	5	4	3	2	1	0
0	0	0	1	コマンド・コード				
1	LUN			—				REL
2	論理ブロック・アドレス (MSB)							
3	論理ブロック・アドレス							
4	論理ブロック・アドレス							
5	論理ブロック・アドレス (LSB)							
6	—							
7	転送サイズ (MSB)							
8	転送サイズ (LSB)							
9	コントロール・バイト						フラグ	リンク

(b) グループ1 CDB

バイト	ビット	7	6	5	4	3	2	1	0
0		1	0	1	コマンド・コード				
1		LUN			—				REL
2		論理ブロック・アドレス(MSB)							
3		論理ブロック・アドレス							
4		論理ブロック・アドレス							
5		論理ブロック・アドレス(LSB)							
6		—							
7		—							
8		—							
9		転送サイズ(MSB)							
10		転送サイズ(LSB)							
11		コントロール・バイト						フラグ	リンク

(c) グループ5 CDB

〈表2〉
コマンド・グループ
CDBのグループ

コマンド・グループ	CDBバイト数
グループ0	6
グループ1	10
グループ2	未定義
グループ3	未定義
グループ4	未定義
グループ5	12
グループ6	定義可能
グループ7	定義可能

有されるため、一度バスを解放して処理が終わった時点でターゲットがバスを占有します。この動作をリセクションと呼びます。

このときのIDは、結合するイニシエータだけです(図9)。

● コマンド・フェーズ

ターゲットにコマンドを送るフェーズです。セクション完了後ターゲットは、コマンド待ちになります。コマンドは、CDB(コマンド記述ブロック)と呼ばれるテーブル形式で送ります。またCDBは、コマンド・グループによってCDBのバイト数が異なります。転送タイミングを図10に示します。このタイミングは、各フェーズ同じであり、コントロール信号のMSG, C/D, I/Oが異なるだけです。

表2にコマンド・グループを示します(図11参照)。

● データ・フェーズ

ターゲットは、コマンドを受け取るとデータ・フェーズに移ります。データ・フェーズは、イニシエータ

〈表3〉
メッセージ・コード

No.	意味
00	コマンド終了
02	セーブ・データ・ポインタ
03	リストア・ポインタ
04	ディスコネクト
06	アボート
07	メッセージ・リジェクト
08	ノー・オペレーション
0A	リンク・コマンド終了
0B	リンク・コマンド終了(フラグ)
0C	バス・デバイス・リセット

とターゲット間でデータの転送を行い、その転送数は、CDB中で指定します。

● ステータス・フェーズ

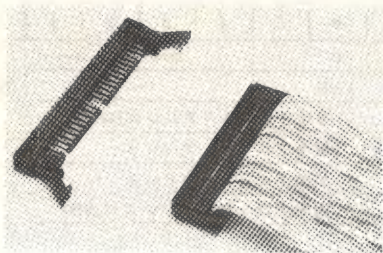
コマンドの終了結果をイニシエータに送るフェーズです。終了結果は、1バイトで示され、正常終了時=00Hです。

● メッセージ・フェーズ

メッセージ・フェーズには、メッセージ・インとメッセージ・アウトがあります。メッセージ・インは、最後のフェーズであり、インターフェース上のメッセージを1バイト分ターゲットから送ります。SCSIでは、デバイスのアクセスに加えてインターフェース上の管理機能もあるからです。表3にメッセージ・コードを示します。

メッセージ・アウト・フェーズは、イニシエータがターゲットに対して要求メッセージを送る場合に使います。セクション・フェーズ中にATNをアクティブにすることによってフェーズを終了後、本フェーズに移行します。1バイトのメッセージを送るとコマンド・フェーズに移行します。

以上のフェーズによって、イニシエータとターゲットのデータ伝送を行います。



§ 4-2

SCSIコントロールLSIの 使い方

里 和政/清水哲夫

最近各メーカからSCSI用のコントローラが発表されていますが、国産の物が少ないようです。筆者は、富士通製のSCSIプロトコル・コントローラ(MB89352)を入手して、これを用いてPC9801用のSCSIボードを製作しました(次章を参照)。

現在入手できるコントロールLSIを表1に示します。

MB89352

MB89352は、1987年に発表された新しいLSIです。この前にMB89351がありますが、これは、SCSIバス・ドライバが外部に必要になります。パッケージも大きめの64ピンDIPです。

● MB89352の特徴

MB89352の機能は、MB89351と同様ですが、SCSIバス・ドライバ/レシーバが内蔵されています。そのためパッケージは、48ピンDIPになっています。

このLSIは、同期転送を除いたSCSI規格をLSI上でサポートしており、ソフトウェアでコマンドを与えることによって、容易にアービトレーション、セクション、リセクションなどのフェーズの実行、またはイニシエータ、ターゲットなどの動作を行うことがで

きます。

また、転送モードは、マニュアル転送、ハード転送(プログラム転送、DMA転送)があります。

マニュアル転送は、SCSI上のACK/REQ信号をソフトによって制御して転送します。プログラム転送は、コントローラのコマンドで行い、DMA転送は、コントローラにDMAモードを設定することで行います。

図1にMB89352のピン配置を、図2に内部ブロック図を示します。表2に各ピンの説明を示します。図3、図4、図5に、リード/ライト、DMAのタイミングを示します。

このコントローラは、14個のレジスタによって制御されます。表3に内部レジスタを示します。

◆ レジスタの説明

レジスタの選択には、 $A_0 \sim A_3$ の4ビットのアドレスで行います。各々のレジスタは、リード/ライトが可能ですが、動作が異なります。

● BDIDレジスタ (0)

ライト時は、SCSI上でのバス・デバイスID(0~7)を2進数で指定します。リード時は、バス・デバイスIDを示します。ビット0~7上の1ビットがONとな

〈表1〉⁽⁹⁾ 現在市販されているSCSI用LSIの主な機能

メーカ名	ウェスタンデジタル	ウェスタンデジタル	日本エヌシーアール	富士通	アダプテック
型 名	WD33C93A	WD33C93	NCR5380S	MB87030/31*	AIC6250
同期転送、非同期転送	可	可	可	可	可
最大転送速度	5Mバイト/秒	4Mバイト/秒	2Mバイト/秒	4Mバイト/秒	5Mバイト/秒
アービトレーション	可	可	可	可	可
転送カウンタ(24ビット)	有	有	有	有	有
ドライバ/レシーバ	内蔵	内蔵	外付け	外付け	内蔵
データ・バッファリング	12バイトFIFO	5バイトFIFO	2バイトFIFO	8バイトFIFO	8バイトFIFO
データ・アクセス・モード	P-I/O DMA バーストDMA WD-バス	P-I/O DMA WD-バス	P-I/O DMA	バーストDMA	P-I/O DMA
オフセット	1~12	1~5	1	1~8	1~8
製造プロセス	C-MOS	C-MOS	NMOS(C-MOS)	C-MOS	C-MOS
パッケージ	40ピンDIP 44ピンPLCC	40ピンDIP 44ピンPLCC	48ピンDIP 44ピンPLCC	88ピンPGA 100ピンQFP	48ピンDIP 52ピンPLCC

* バス・ドライバがつき改良されたのがMB89352。詳しくは§4-2に解説。新たにMB87033も発表された

〈表 2〉⁽¹⁾ MB89352のピン機能

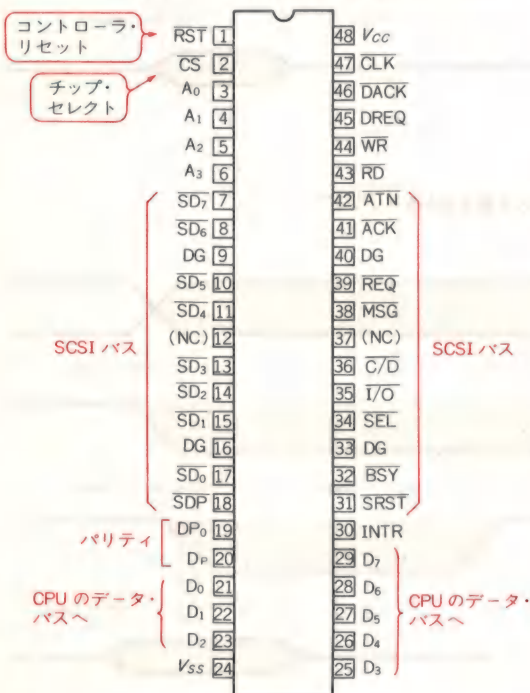
信号名	入出力	ピン番号	機 能	信号名	入出力	ピン番号	機 能
RST	入 力	1	コントローラのリセット入力	DACK	入 力	46	DMA要求に対する応答信号
CS	入 力	2	コントローラのセレクト入力	DREQ	出 力	45	DMA転送の要求信号
A ₀	入 力	3	コントローラの内部レジスタをセレクトするためのアドレス入力	SD ₇	入出力	7	SCSI上のデータ・バス 負論理信号
A ₁		4		SD ₆		8	
A ₂		5		SD ₅		10	
A ₃		6		SD ₄		11	
DP ₀	出 力	19	D ₇ ～D ₀ の奇数パリティ出力	SD ₃		13	
D _P	入 力	20	奇数パリティ・ビット。 8ビットの双方向性3ステートのデータ・バス。 RD, WRが“H”のときハイ・インビ ーダンスになる	SD ₂		14	
D ₇		29		SD ₁		15	
D ₆		28		SD ₀		17	
D ₅		27		SD _P		18	
D ₄		26		SEL	入出力	34	SCSI上の制御信号であり、負論理
D ₃		25		BSY		32	
D ₂		23		I/O		35	
D ₁		22		C/D		36	
D ₀		21		MSG		38	
RD	入 力	43	内部レジスタを読み出すためのス トロープ信号, CS=“L”で有効となる	REQ		39	
WR	入 力	44	内部レジスタを書き込むためのス トロープ信号, CS=“L”で有効となる	ACK		41	
CLK	入 力	47	コントローラの制御のためのクロ ック	ATN		42	
INTR	出 力	30	コマンドの終了, またはエラーが発 生したことを通知するための信号	SRST		31	
				V _{CC}	入 力	48	+5V電源入力
				V _{SS}		24	0V (GND)
				DG		9	ドライバ・グラウンド, V _{SS} と同等
						16	
						33	
						40	

ります。

● SCTLレジスタ (1)

コントローラの動作モードを設定します。コントロ

〈図 1〉⁽¹⁾ MB89352のピン配置



ーラのリセット, 各フェーズの動作を決定します(図 6 参照)。

● SCMDレジスタ (2)

ライトされたコマンドを実行します。またはコントローラとCPUとの転送モードを設定します。

コマンドを書き込みコントローラが実行可能な状態であれば, そのコマンドを実行します(図 7)。

以下のコマンドがあります。

(1) バス・リリース

SCSIバスを解放します。

(2) セレクト

セレクション, リセクションを行います。

(3) リセットATN

ATN信号をリセットします。

(4) セットATN

ATN信号をセットします。

(5) トランスファ

データの転送を行います。

(6) トランスファ・ポーズ

ターゲット動作のとき, ハード転送を一時停止させます。

(7) リセットACK/REQ

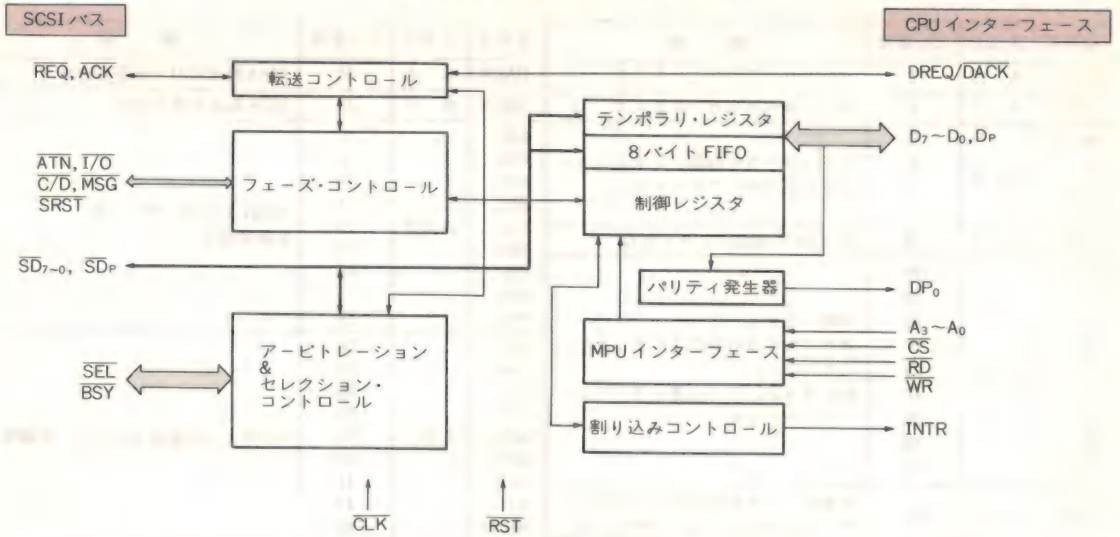
ACK/REQ信号をリセットします。

(8) セットACK/REQ

ACK/REQ信号をセットします。

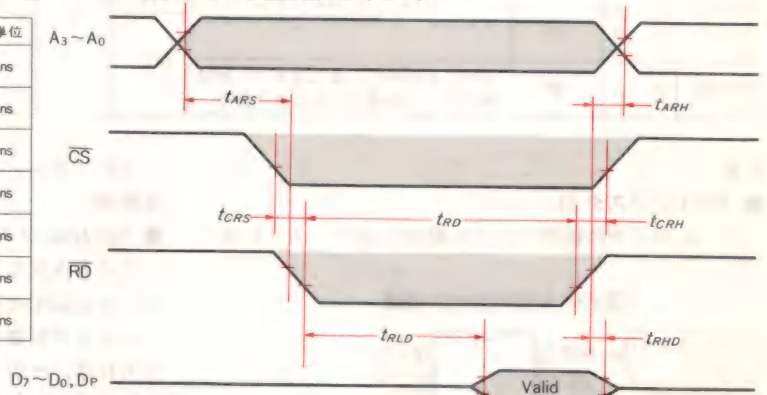
● INTSレジスタ (4)

〈図2〉⁽¹⁾ MB89352の内部ブロック図



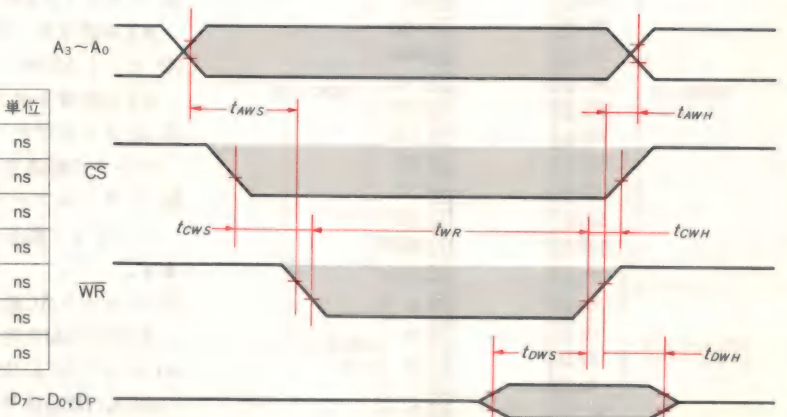
〈図3〉⁽¹⁾ MB89352のレジスタ読み出しタイミング

項 目	記号	条 件	最小	最大	単位
アドレス・セットアップ	t_{ARS}		40		ns
アドレス・ホールド	t_{ARH}		10		ns
CS セットアップ	t_{CRS}		25		ns
CS ホールド	t_{CRH}		10		ns
RD パルス幅	t_{RD}		120		ns
RD "L"→データ確定	t_{RLD}	負荷容量 $C_L=80\text{pF}$	90		ns
RD "H"→データ・ホールド	t_{RHD}	負荷容量 $C_L=20\text{pF}$	10	60	ns

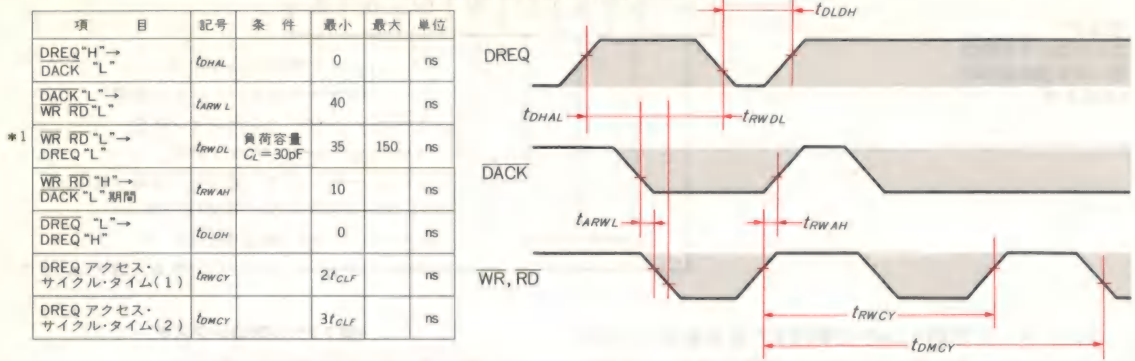


〈図4〉⁽¹⁾ MB89352のレジスタ書き込みタイミング

項 目	記号	最小	単位
アドレス・セットアップ	t_{AWS}	40	ns
アドレス・ホールド	t_{AWH}	10	ns
CS セットアップ	t_{CWS}	25	ns
CS ホールド	t_{CWH}	10	ns
データ・バス・セットアップ	t_{DWS}	30	ns
データ・バス・ホールド	t_{DWH}	20	ns
WR パルス幅	t_{WR}	100	ns



〈図5〉⁽¹⁾ MB89352のDMAアクセスのタイミング



- (※1) ライト時(アウトプット時), データ・バッファ・レジスタがFull になるときに適用する。
 リード時(インプット時), データ・バッファ・レジスタがEmpty になるときに適用する。
 (※2) WR, RD パルス幅は, レジスタ書き込みタイミング, 読み出しタイミングの規定に従う。

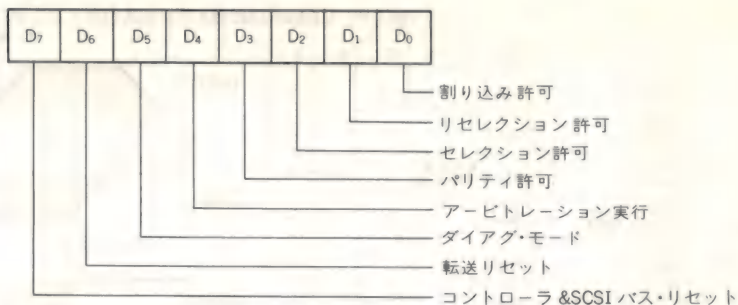
〈表3〉⁽²⁾ 内部レジスタ

アドレス (16進数)	名 称 (略称)	R W	レ ジ ス タ ・ ビ ッ ト										バリテイ
			ビット7	ビット6	ビット5	ビット4	ビット3	ビット2	ビット1	ビット0			
0	Bus Device ID (BDID)	R	# 7	# 6	# 5	# 4	# 3	# 2	# 1	# 0	"0"		
		—						ID ₄	ID ₂	ID ₁	—		
1	Spc Control (SCTL)	R	Reset & Disable	Control Reset	Diag Mode	ARBIT Enable	Parity Enable	Select Enable	ReSelect Enable	INT Enable	P		
		—					RST Out	Intercept Xfer	Transfer Modifier	PRG Xfer	"0"	Term Mode	
2	Command (SCMD)	R	Command Code			RST Out	Intercept Xfer	Transfer Modifier			P		
		—					Command Complete	Service Required	Time Out	SPC Hard Error	Reset Condition		
4	Interrupt Sense (INTS)	R	Selected	Re-Selected	Dis-Connected	Command Complete	Service Required	Time Out	SPC Hard Error	Reset Condition	P		
		Reset Interrupt										—	
5	Phase Sense (PSNS)	R	REQ	ACK	ATN	SEL	BSY	MSG	C/D	I/O	P		
	SPC Diag. Control (SDGC)	W	Diag REQ	Diag ACK	Xfer Enable	—	Diag BSY	Diag MSG	Diag C/D	Diag I/O	—		
6	SPC Status (SSTS)	R	Connected INIT TARG		SPC BSY	Xfer in Progress	SCSI RST	TC=0	DREG FULL	Status EMPTY	P		
7	SPC Error Status (SERR)	R	Data SCSI	Error SPC	Xfer Out	"0"	TC P-Error	"0"	Short Period	"0"	P		
8	Phase Control (PCTL)	R	Bus Free Interrupt Enable	"0"				Transfer MSG Out	Phase C/D Out	I/O Out	P		
		—						MBC					
9	Modified Byte Counter (MBC)	R	"0"				ビット3	ビット2	ビット1	ビット0	P		
A	Data Register (DREG)	R	Internal Data Register (8 バイト FIFO)										P
		ビット7	ビット6	ビット5	ビット4	ビット3	ビット2	ビット1	ビット0	—			
B	Temporary Register (TEMP)	R	Temporary Data (Input : From SCSI)										P
		ビット7	ビット6	ビット5	ビット4	ビット3	ビット2	ビット1	ビット0	—			
C	Transfer Counter High (TCH)	R	Transfer Counter High (MSB)										P
		ビット23	ビット22	ビット21	ビット20	ビット19	ビット18	ビット17	ビット16	—			
D	Transfer Counter Mid (TCM)	R	Transfer Counter Middle (2nd Byte)										P
		ビット15	ビット14	ビット13	ビット12	ビット11	ビット10	ビット9	ビット8	—			
E	Transfer Counter Low (TCL)	R	Transfer Counter Low (LSB)										P
		ビット7	ビット6	ビット5	ビット4	ビット3	ビット2	ビット1	ビット0	—			

- (注1) 書き込み動作において一か記入されているビットは, "1", "0" のどちらかを書き込んでもよいことを示す。
 (注2) 読み出し専用レジスタへの書き込み動作は無視される。
 (注3) 読み出し時において "0" と示されているビットは必ず "0" が読み出される。

〈図6〉⁽²⁾

コントローラの動作
モードを決めるSCT
Lレジスタ



コントローラの割り込みの要因または解除を行います。コントローラから割り込みが必要な場合、INTR信号を出力しますが、SCTLレジスタによりマスク可能です。割り込み解除は、割り込み要因の対応するビットに“1”を書き込むことにより行います。

割り込み原因には、SCSIバス上でセクション、リセクションが行われたとき、セレクト、転送コマンドが終了したときなどがあります。

本レジスタをリセットしないで、コマンドを発行しても正常に動作は行われません。

● PSNSレジスタ (5)

ダイアグ・モード(テスト・モード)の場合、SCSI上の制御線をコントロールします。それ以外では、SCSI上の制御信号の状態を示します。

これによってSCSIの制御線の確認ができます。このレジスタは、コントローラの状態にかかわらずリードできます。

● SDGCレジスタ (5)

ダイアグ・モードのとき、コントローラを疑似的にコントロールするレジスタです。

● SSTSレジスタ (6)

コントローラの内部状態を示すレジスタで、常にリード可能です。

● SERRレジスタ (7)

コントローラ内部で検出されたエラーの内容を示します。

● PCTLレジスタ (8)

実行する転送フェーズを指定します。またはセクション、リセクションの区別を行います。

● MBCレジスタ (9)

プログラムまたはDMA転送モードでの、データ転送バイトを示すカウンタです。

● DREGレジスタ (10)

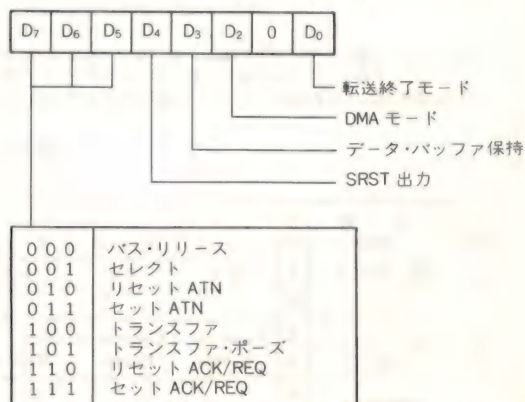
コントローラ内部の転送用データ・レジスタです。8バイトのFIFOから構成されています。ハード転送の場合このレジスタを使用します。

● TEMPレジスタ (11)

ハード転送以外で転送する場合に使用します。

● TCH, TCM, TCLレジスタ (12, 13, 14)

〈図7〉⁽²⁾ SCMDレジスタ



3バイトから成るレジスタで、ハード転送時カウンタ・ダウンされて、転送の残りバイトを示します。それ以外に、セレクト時のタイム・アウト値を設定します。

◆ 動作の説明

コントローラは、これらのレジスタによってSCSIのコントロールを行います。そのコントロールは、SCMDレジスタのコマンドで行います。

● バス・フリー

ターゲット動作中に、バス・フリー・フェーズへの移行を要求します。またバス・フリー待ちのセレクト・コマンドを解除します。

● セレクト

イニシエータ、ターゲットの結合を開始します。もしバス・フリー状態でなければ、バス・フリーが検出されるのを待ちます。

SCTLレジスタのアービトレーション=1の場合は、アービトレーション・フェーズを行い、バスの占有権を獲得します。バス獲得ができなかったとき、コマンドは終了します。

アービトレーション=0の場合は、すぐにセクションを開始します。セクションは、PCTLレジスタのビット0(I/O)=0のときセクション・フェーズを、ビット0=1のときリセクション・フェーズを

実行します。

セクション・フェーズにおいてANT信号が必要なときは、SET ANTコマンドをあらかじめ発行しておきます。

TEMPレジスタには、結合するイニシエータまたはターゲットのID番号を入れておきます。

TCH, TCMレジスタは、BSY信号の時間監視の値をセットします。この時間は、セクション・リセクション・フェーズ開始してから、BSY信号がアクティブになるまでです。

N (TCH : High, TCM : Low)

$N \neq 0$ の場合 時間 = $(N \times 256 + 15) \times T_{clk} \times 2$

$N = 0$ の場合 時間 = ∞

(T_{clk} はコントローラのクロック周期)

TCLレジスタは、バス・フリー後アービトレーション・セクション・フェーズを開始するための時間をセットします。値は0～15までの値です。

時間 = $(TCL + 6) \times T_{clk} \sim (TCL + 7) \times T_{clk}$

コントローラのクロックが8 MHzのとき、TCL = 4で1.25 μ sとなります。

● セット ATN

イニシエータとして動作する場合のみ有効となります。

結合前にコマンドが出された場合は、セクション・フェーズ実行時にATN信号がアクティブになります。結合中であればコマンドが出された時点で、ATN信号がアクティブとなります。

● リセット ATN

アクティブのATN信号をインアクティブにします。ただし、SCSIのバス結合が解除された場合、本コマンドが発行しなくてもリセットされます。

● トランスファ

各転送フェーズを実行させます(コマンド、データ、ステータス、メッセージ・フェーズ)。

このコマンドで実行される転送モードは、ハード転送と呼び、コントローラによって転送シーケンスの制御が行われます。転送は、レジスタによって制御します。

TCH, TCM, TCLレジスタは、転送バイト数を24ビットで指定します。

PCTLレジスタは、転送フェーズを3ビットで指定します。

000 : データ・アウト・フェーズ

001 : データ・イン・フェーズ

010 : コマンド・フェーズ

011 : ステータス・フェーズ

110 : メッセージ・アウト・フェーズ

111 : メッセージ・イン・フェーズ

コントローラは、PCTLのフェーズとバス実行フェーズが同じとき転送を開始します。もしフェーズが一致しない場合は、サービス要求割り込みが発生します。

このときは、フェーズを合わせて再度転送を行うか、マニュアルで転送を行います。

● トランスファ ポーズ

ターゲット動作しているとき、ハード動作を中止させます。

● セット ACK/REQ

マニュアル転送を行うためコマンドです。イニシエータ動作時は、ACK信号をターゲット動作時は、REQ信号をアクティブにします。

マニュアル転送のときは、TEMPレジスタでデータのやり取りをします。ハード転送と同様にPCTLレジスタに転送フェーズをセットしておきます。

● リセット ACK/REQ

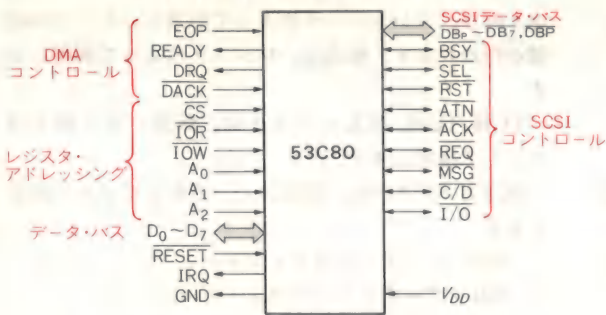
セット ACK/REQコマンドでアクティブにした信号をリセットします。

イニシエータ動作時はACK信号を、ターゲット動作時はREQ信号をインアクティブにします。

〈表4〉 富士通製SCSIコントローラ一覧

	MB87033	MB87030/31	MB89352	MB89351
シングルエンド・ドライバ/レシーバ	内蔵	外付け	内蔵	外付け
ディファレンシャル対応	不可	可	不可	可
同期転送	可	可	不可	不可
転送バイト・カウンタ	28ビット	24ビット	24ビット	24ビット
アービトレーション失敗時割り込み	あり	なし	なし	なし
アテンション・コンディション検出割り込み	あり	なし	なし	なし
FIFO Full/Empty 割り込み	なし	なし	あり	あり
割り込み信号数	2	1	1	1
CPUバス・パリティ・ジェネレータ	あり	なし	あり	あり
データ転送バス	独立	独立	CPUバスと共通	CPUバスと共通

〈図8〉機能別ピン配置



〈図9〉⁽¹⁾ピン配置

DB ₇	1	48	DB ₆
RST	2	47	DB ₅
GND	3	46	GND
BSY	4	45	DB ₄
SEL	5	44	DB ₃
ATN	6	43	DB ₂
NC	7	42	NC
RESET	8	41	DB ₁
IRQ	9	40	DB ₀
DRQ	10	39	GND
EOP	11	38	DB _P
DACK	12	37	REQ
GND	13	36	ACK
READY	14	35	I/O
A ₀	15	34	GND
A ₁	16	33	C/D
A ₂	17	32	MSG
NC	18	31	NC
CS	19	30	D ₀
IOW	20	29	D ₁
IOR	21	28	D ₂
D ₇	22	27	D ₃
D ₆	23	26	D ₄
D ₅	24	25	V _{DD}

なお、富士通製のSCSIコントローラの比較を表4に示します。

NCR53C80

次に、市場に出るのが早く、利用の多いNCRの53C80について説明します。

図8に機能別に分けたピン配置、図9に53C80のピン配置、表5に各ピンの詳しい説明を示します。また、図10に内部ブロック図を示します。

内部レジスタ

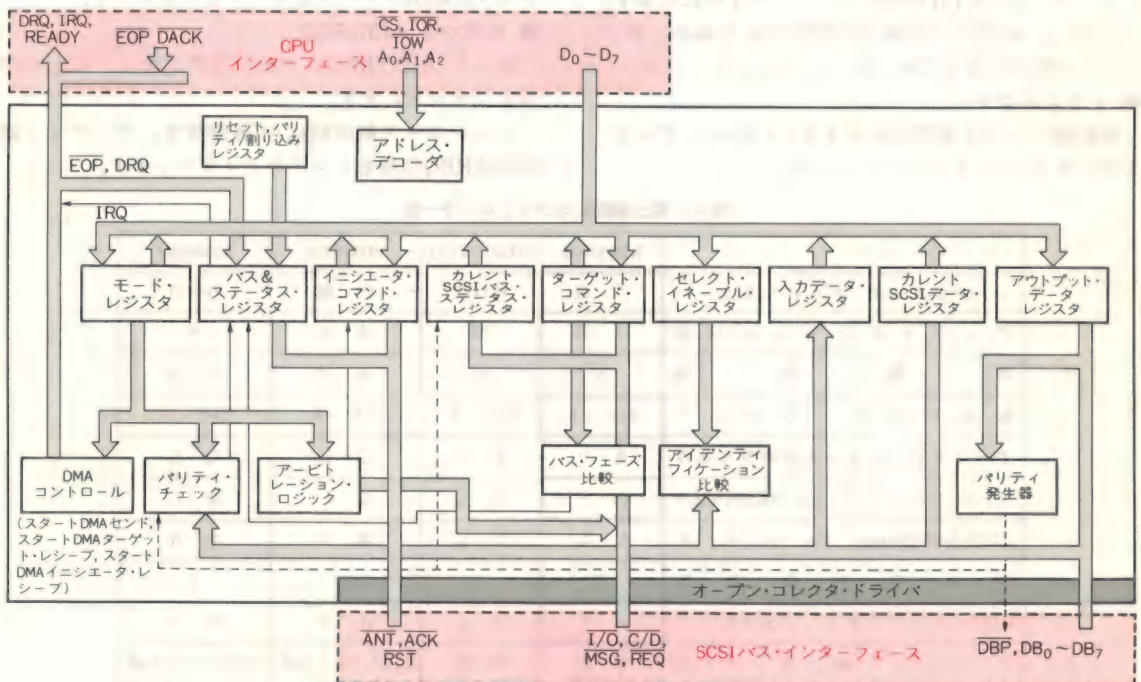
53C80は、CPUから見ると八つのレジスタの集合体としてみることができます。プログラミングは、この八つのレジスタを駆使して行うことになります(表6)。

● カレントSCSIデータ・レジスタ (R) アドレス = 0

プログラムI/Oで、CPUがSCSIデータ・バスを読み込むとき、アービトレーション中に、ほかのプライオリティの高いデバイスが、アービトレーションを行っていないかどうかをチェックするときに使用します。

モード・レジスタで、イネーブル・パリティ・チェック・ビットが1にセットされているときは、このレジスタのリード・サイクルの最初で、パリティがチェックされます。パリティ・エラーのときは、バス/ステータス・レジスタのパリティ・エラー・ビットが1に

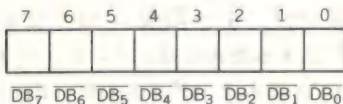
〈図10〉⁽¹⁾ 53C80の内部ブロック図



〈表5〉53C80のピン説明

信号名	ピン番号	入出力	説 明	信号名	ピン番号	入出力	説 明			
DMAインターフェース				D ₇	22	入出力 3ステ ート	データ・バス			
$\overline{\text{EOP}}$	11	入力	End Of Process. DMA転送中に $\overline{\text{EOP}} = \text{"L"}$ とすると、 DMA転送が終了する	D ₆	23					
READY	14	出力	ブロック・モードDMAでの1バイト ごとのDMA要求を表す	D ₅	24					
DRQ	10	出力	DMAリクエスト。モード・レジスタ のDMAモード・ビット = 1 で、データ・ レジスタ中にデータが用意されたとき に出力される。 ノーマル・モードのDMAでは、DMA/ DACKで1バイトごとのハンドシェ イクを行う	D ₄	26					
				D ₃	27					
$\overline{\text{DACK}}$	12	入力	DMAアクノレッジ。 $\overline{\text{DACK}} = \text{"L"}$ に よってDRQがリセット(= "H")され、 データ・レジスタが入出力のために セレクトされる	D ₂	28					
				D ₁	29					
				D ₀	30					
電源				IRQ	9	出力	インタラプト・リクエスト			
V _{DD}				25		+ 5 Vサブライ	$\overline{\text{RESET}}$	8	入力	すべての内部レジスタをクリアする (ただし、SCSIバス上には、 $\overline{\text{RST}}$ は 出力しない)
GND				3,13,34, 39,46		グラウンド。ノイズ・マージン改善 のために、5本のグラウンドが用意 されている	SCSIインターフェース			
CPUインターフェース				ATN	6	SCSIバス信号				
$\overline{\text{CS}}$				19	入力		チップ・セレクト。A ₂ ~A ₀ で指定さ れた内部レジスタのリード/ライトを イネーブルする	BSY	4	
$\overline{\text{IOR}}$				21	入力		I/Oリード・ストロープ。 $\overline{\text{CS}}$ と A ₂ ~A ₀ で指定されたレジスタを読み込む。 また、 $\overline{\text{DACK}}$ と共に使われると ($\overline{\text{DACK}}$ = "L", $\overline{\text{IOR}} = \text{"L"}$)、インプット・デ ータ・レジスタがセレクトされる (DMA転送)	ACK	36	
								RST	2	
$\overline{\text{IOW}}$	20	入力	I/Oライト・ストロープ。 $\overline{\text{CS}}$ と A ₂ ~A ₀ で指定されたレジスタに書き込む。 また、 $\overline{\text{DACK}}$ と共に使われると ($\overline{\text{DACK}}$ = "L", $\overline{\text{IOW}} = \text{"L"}$)、アウトプット・ データ・レジスタがセレクトされる (DMA転送)	MSG	32					
				SEL	5					
A ₂	17	入力	$\overline{\text{CS}}$, $\overline{\text{IOR}}$, $\overline{\text{IOW}}$ と共に使われ、内部 レジスタのセレクトを行う	C/D	33					
				REQ	37					
A ₁	16	入力		I/O	35					
A ₀	15			DB ₇	1					
				DB ₆	48					
				DB ₅	47					
				DB ₄	45					
				DB ₃	44					
				DB ₂	43					
				DB ₁	41					
				DB ₀	40					
				DB _P	38					

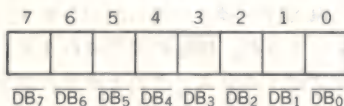
セットされます。アービトラージョン中には、パリティ
ィ・ビットは常にフォールスです。



(カレントSCSIデータ・レジスタ)

● アウトプット・データ・レジスタ (W) アドレス=0
プログラムI/O, DMA I/Oの両方で、SCSIバスに
データを出力するときに使用します。

プログラムI/Oのときは、 $\overline{\text{CS}} = \text{"L"}$, A₂ = "L",
A₁ = "L", A₀ = "L", $\overline{\text{IOW}} = \text{"L"}$ でセレクトされ、
DMA I/Oのときは、 $\overline{\text{DACK}} = \text{"L"}$, $\overline{\text{IOW}} = \text{"L"}$,
 $\overline{\text{CS}} = \text{"H"}$, A₂~A₀ = × でセレクトされます。また、
アービトラージョン, セレクション, リセレクション
時のIDビットを出力するときににも使用されます。



(アウトプット・データ・レジスタ)

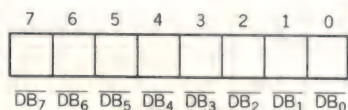
〈表6〉53C80の内部レジスタ

A ₂	A ₁	A ₀	R/W	レ ジ ス タ 名
0	0	0	R W	カレントSCSIデータ・レジスタ アウトプット・データ・レジスタ
0	0	1	R/W	イニシエータ・コマンド・レジスタ
0	1	0	R/W	モード・レジスタ
0	1	1	R/W	ターゲット・コマンド・レジスタ
1	0	0	R W	カレントSCSIバス・ステータス・レジスタ セレクト・イネーブル・レジスタ
1	0	1	R W	バス/ステータス・レジスタ スタートDMA送・レジスタ
1	1	0	R W	インプット・データ・レジスタ スタートDMAターゲット・レシーブ・レジスタ
1	1	1	R W	リセット・パリティ/インタラプト/レジスタ スタートDMAイニシエータ・レシーブ・レジスタ

● インプット・データ・レジスタ (R) アドレス=6
DMA I/Oで、SCSIバス上のデータをラッチします。
 $\overline{\text{IOR}} = \text{"L"}$, $\overline{\text{DACK}} = \text{"L"}$, $\overline{\text{CS}} = \text{"H"}$, A₂
~A₀ = × でセレクトされます。また、CPUからも、
 $\overline{\text{IOR}} = \text{"L"}$, $\overline{\text{CS}} = \text{"L"}$, A₂ = "H", A₁ = "H",

$A_0 = "L"$ でセレクトすることができます。

このレジスタにデータがラッチされるのは、モード・レジスタでDMAモードに設定され、DMAターゲット・レシーブ中の、 \overline{ACK} の立ち下がり(┐)と、DMAイニシエータ・レシーブ中の、 \overline{REQ} の立ち下がり(┐)のときです。

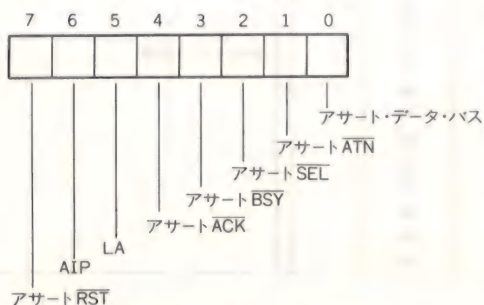


[インプット・データ・レジスタ]

● イニシエータ、コマンド・レジスタ (R/W) アドレス = 1

イニシエータに関係の深いSCSIバス信号(\overline{ACK} , \overline{BSY} , \overline{SEL} , \overline{ATN} , \overline{RST})をアサート(出力)したり、アービトレーションの状態をモニタするときに使します。

● 読み込み時



リード時に重要なのは、LAビットとAIPビットの二つで、ほかのビットは、たんにそのビットの状態をリード・バックしているにすぎません。

▶ LA (Lost Arbitration) ビット

アービトレーションに負けたかどうかをモニタします。このビットは、モード・レジスタのアービトレーション・ビットに1が書き込まれてから意味をもつようになり、LA = 1であるということは、次の状態を意味します。

- (1) バス・フリーを検出し、
- (2) $\overline{BSY} = "L"$ にし、
- (3) SCSIデータ・バスにIDを出力したが、
- (4) ほかのデバイスが、 $\overline{SEL} = "L"$ にしたので、アービトレーションに負けた。

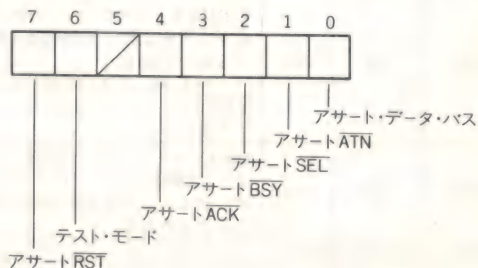
▶ AIP (Arbitration In Progress) ビット

アービトレーションが、進行中かどうかをモニタします。このビットもモード・レジスタのアービトレーション・ビットに1が書き込まれてから意味をもつようになり、AIP = 1であるということは、次の状態を意味します。

- (1) バス・フリーが検出され、
- (2) $\overline{BSY} = "L"$ にし、
- (3) アウトプット・データ・レジスタの内容をSCSIデータ・バスに出力している。

この状態は、モード・レジスタのアービトレーション・ビットがリセットされるまで変化しません。

● 書き込み時



▶ ビット 7 = アサート \overline{RST}

1を書き込むと、SCSIバス上の $\overline{RST} = "L"$ となります。この状態($\overline{RST} = "L"$)は、このビットに0を書くか、またはチップがリセット($\overline{RESET} = "L"$)されるまで、解除されません。

同時にIRQ(9ピン)もアサートされ、インタラプト・ラッチと、このビットを除くすべての内部ロジック、コントロール・レジスタはリセットされます。

▶ ビット 6 = テスト・モード

チップのメンテナンス/テスト用のビットで、1を書き込むと、すべての出力ドライバをディセーブルします。0にもどすと、通常のオペレーションになります。

▶ ビット 4 = アサート \overline{ACK}

1で $\overline{ACK} = "L"$ 、0で $\overline{ACK} = "H"$ にします。 $\overline{ACK} = "L"$ にするためには、イニシエータとして(モード・レジスタのターゲット・モード・ビット = 0)、設定されていなければなりません。

▶ ビット 3 = アサート \overline{BSY}

1で $\overline{BSY} = "L"$ 、0で $\overline{BSY} = "H"$ にします。

▶ ビット 2 = アサート \overline{SEL}

1で $\overline{SEL} = "L"$ 、0で $\overline{SEL} = "H"$ にします。

▶ ビット 1 = アサート \overline{ATN}

1で $\overline{ATN} = "L"$ 、0で $\overline{ATN} = "H"$ にします。ビット4と同様 $\overline{ATN} = "L"$ にするためには、イニシエータとして設定されていることが必要です。

▶ ビット 0 = アサート・データ・バス

1を書き込むと、アウトプット・データ・レジスタの内容を、SCSIデータ・バスに出力します。パリティもジェネレートされ、 \overline{DB}_p に出力されます。

このチップがイニシエータのときは、以下の条件が必要で、

(1) モード・レジスタのターゲット・モード・ビット = 0

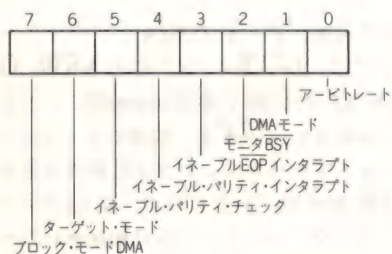
(2) SCSIバスの $\overline{I/O}$ = “H”

(3) ターゲット・コマンド・レジスタ中の下位 3 ビットが、SCSIバス上の、 $\overline{C/D}$, $\overline{I/O}$, \overline{MSG} にそれぞれマッチしていること

また、DMAセンドのときも、このビットが 1 に設定されていなければなりません。

● モード・レジスタ (R/W) アドレス = 2

チップのモードを設定します。



〔モード・レジスタ〕

▶ ビット 7 = ブロック・モード DMA

DMAのDRQ/DACKのハンドシェイクの方法を設定します。DMAモード・ビットが 1 で、このビットが 0 のときは、通常のインターロック・ハンドシェイクを行い、1 バイトごとに、DRQと \overline{DACK} がハンドシェイクを行います。

DMAモード・ビットが 1 で、このビットも 1 のときには、ブロック・モードDMAと呼ばれ、 \overline{DACK} は複数バイトにわたって常に“L”です。 \overline{IOR} または \overline{IOW} のパルスで、1 バイトの転送が行われ、READYピン(ピン14)が、次の1 バイトの転送要求を行います。

▶ ビット 6 = ターゲット・モード

1 でターゲット、0 でイニシエータとしてこのチップを設定します。 \overline{ATN} , \overline{ACK} をアサートするときは 0, $\overline{C/D}$, $\overline{I/O}$, \overline{MSG} , \overline{REQ} をアサートするときは 1 にそれぞれ設定します。

▶ ビット 5 = イネーブル・パリティ・チェック

パリティ・エラーが発生したときに、その情報をバス/ステータス・レジスタのパリティ・エラー・ビットにセーブするか、無視するかを決めます。1 でセーブ、0 で無視です。

▶ ビット 4 = イネーブル・パリティ・インタラプト

1 にセットすると、パリティ・エラーが発生したときに、インタラプト (IRQ) を発生します。このインタラプトを発生させるには、ビット 5 のイネーブル・パリティ・チェック・ビットも 1 にセットしておかなければなりません。

▶ ビット 3 = イネーブル EOP インタラプト

1 にセットすると、DMACからの \overline{EOP} (ピン11) =

“L” で、インタラプトを発生します。

▶ ビット 2 = モニタ BSY

1 にセットすると、SCSIバス上の \overline{BSY} が“H”になってはいけな所“H”になったときに、インタラプトを発生します。このインタラプトが発生すると、イニシエータ・コマンド・レジスタの下位 6 ビットがクリアされ、すべての信号がSCSIバスから取り除かれます。

▶ ビット 1 = DMAモード

DMA転送を行う前にセットすべきもので、スタートDMAセンド・レジスタ、スタートDMAターゲット・レシープ・レジスタ、スタートDMAイニシエータ・レシープ・レジスタに書き込む前に設定します。また、ターゲット・モード・ビットも、スタートDMAイニシエータ・レシープを行うときには 0, スタートDMAターゲット・レシープを行うときには 1 にセットします。

さらに、スタートDMAセンドを行うときには、イニシエータ・コマンド・レジスタのアサート・データ・バス・ビットを 1 にセットしておかなければなりません。なお、このビットは、SCSIバスで \overline{BSY} = “L”になっているときでないとセットできません。

《重要》

DMAモードでは、REQ/ACKのハンドシェイクは、自動的に行われます。また、 \overline{CS} と \overline{DACK} が同時に“L”になることは許されません。

▶ ビット 0 = アービトレート

アウトプット・データ・レジスタにIDを書いたのち、このビットを 1 にセットすると、次のプロセスを行います。

(1) バス・フリーが検出されるまで待つ

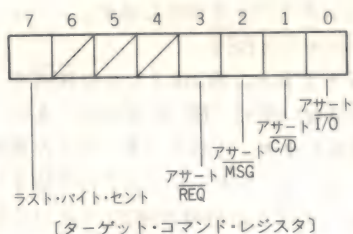
(2) IDを出力し、 \overline{BSY} = “L” とする

(3) アービトレーションの結果を、イニシエータ・コマンド・レジスタのAIP, LA各ビットにセットする

● ターゲット・コマンド・レジスタ (R/W) アドレス = 3

ターゲットとして接続された場合(モード・レジスタのターゲット・モード・ビット = 1), ターゲットの動作を行う信号($\overline{I/O}$, $\overline{C/D}$, \overline{MSG} , \overline{REQ})をコントロールします。また、イニシエータとして接続した場合は、データをSCSIバスに出力するためには、このレジスタの下位 3 ビットは、カレントSCSIバス・ステータス・レジスタの \overline{MSG} , $\overline{C/D}$, $\overline{I/O}$ の各ビットと一致していなければなりません。

さらに、イニシエータでかつ、DMAモードのときは、SCSIバス上の \overline{MSG} , $\overline{C/D}$, $\overline{I/O}$ の各信号が、これらのビットと一致していないと、 \overline{REQ} = “L” になったときに、フェーズ・ミスマッチのインタラプトが発生します。

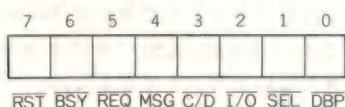


▶ビット7=ラスト・バイト・セント (R)

DMAで、最後のバイトがSCSIバス上に出力され、それがREQ/ACKによって転送され終わったときに、セットされます。

● カレントSCSIバス・ステータス・レジスタ (R) アドレス=4

七つのSCSIバスの制御信号線と、パリティ・ビットをモニタします。現在のバスのフェーズがどうなっているか、REQを出しているターゲットがいないか、また、インタラプトが発生したときに、何が原因かをチェックする目的で使用されます。

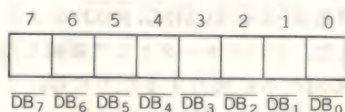


[カレントSCSIバス・ステータス・レジスタ]

● セレクト・イネーブル・レジスタ (W) アドレス=4

セクション/リセクション・フェーズで、自分のIDがセレクト/リセレクトされたときに、インタラプトが発生できます。すなわち、このレジスタに書き込んだIDビットと同じIDがSCSIバスに現れ、かつ、 $\overline{\text{BSY}} = \text{"H"}$ 、 $\overline{\text{SEL}} = \text{"L"}$ という状態が、最低400ns以上続いたときにインタラプトが発生します。このインタラプトは、0を書き込むことでディセーブルできます。

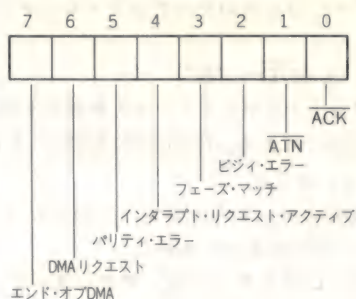
なお、モード・レジスタのイネーブル・パリティ・チェック・ビットが1にセットされている場合は、セクション/リセクション中にもパリティ・ビットがチェックされます。



[セレクト・イネーブル・レジスタ]

● バス/ステータス・レジスタ (R) アドレス=5

ここでは、カレントSCSIバス・ステータス・レジスタに入りきらなかったSCSIバスの制御信号線(ATN, ACK)と、その他のステータスをモニタすることができます。



[バス/ステータス・レジスタ]

▶ビット7=エンド・オブ・DMA

$\overline{\text{EOP}} = \text{"L"}$ 、 $\overline{\text{DACK}} = \text{"L"}$ のとき $\overline{\text{IOR}}$ 、 $\overline{\text{IOW}}$ のどちらかが "L" の状態が最低100ns続いたときに、1にセットされます。 $\overline{\text{EOP}}$ は、最後のデータ・バイトがアウトプット・データ・レジスタに書き込まれたときに、DMACがアサートするので、SCSIバス上で、その最後のデータ・バイトが本当に転送されたかどうかは、REQとACKの信号をモニタして確かめなければなりません。

このビットは、モード・レジスタのDMAモード・ビットが0のときには常に0です。

▶ビット6=DMAリクエスト

DRQ(ピン10)の状態をCPUがモニタするときに使用できます。DRQ信号は、 $\overline{\text{DACK}} = \text{"L"}$ にするか、モード・レジスタのDMAモード・ビットを0にすることで、クリアされます。また、フェーズ・ミスマッチ・インタラプトが発生したときには、DRQはリセットされません。

▶ビット5=パリティ・エラー

このビットは、モード・レジスタのイネーブル・パリティ・チェック・ビットが1にセットされ、データ・レシブ中およびセクション/リセクション中にパリティ・エラーが発生したときに、1にセットされます。リセット・パリティ/インタラプト・レジスタを読み込むことによって、クリアされます。

▶ビット4=インタラプト・リクエスト・アクティブ

インタラプトが発生していることを示すビットで、IRQ(ピン9)の状態をモニタします。リセット・パリティ/インタラプト・レジスタを読み込むことでクリアされます。

▶ビット3=フェーズ・マッチ

SCSIバス上のMSG, C/D, I/Oの状態と、ターゲット・コマンド・レジスタの下位3ビットの状態が一致していると、1にセットされます。このビットは、常にアップデートされており、チップがイニシエータとして設定されているときに意味をもちます。

▶ビット2=ビジー・エラー

モード・レジスタのモニタBSYビットが1にセット

されているとき、BSYが“H”になってはいけな
いときに“H”になると、1にセットされます。この
エラーが発生すると、すべてのSCSIバスへの出力ドラ
イバがディセーブルされ、モード・レジスタのDMA
モード・ビットがクリアされます。

● スタートDMAセンド・レジスタ (W) アドレス＝
5

DMAセンド動作(DMA→SCSIバス)を開始します。
イニシエータ、ターゲットの両方で使用されます。こ
のレジスタにライト・アクセスする前に、モード・レ
ジスタのDMAモード・ビットが1にセットされてい
なければなりません。なお、書き込む値は意味をもち
ません。

● スタートDMAターゲット・レシーブ・レジスタ
(W) アドレス＝6

ターゲットが、DMAレシーブ動作(SCSIバス→
DMA)を開始します。

このレジスタにライト・アクセスする前に、モー
ド・レジスタのDMAモード・ビットおよびターゲッ

ト・モード・ビットの両方が、共に1にセットされて
いなければなりません。書き込む値は意味をもちま
せん。

● スタートDMAイニシエータ・レシーブ・レジスタ
(W) アドレス＝7

イニシエータが、DMAレシーブ動作(SCSIバス→
DMA)を開始します。このレジスタにライト・アクセ
スする前に、モード・レジスタのDMAモード・ビッ
ト＝1、ターゲット・モード・ビット＝0に設定され
ていなければなりません。書き込む値は意味をもちま
せん。

● リセット・パリティ/インタラプト・レジスタ (R)
アドレス＝7

このレジスタにリード・アクセスすると、バス/ステ
ータス・レジスタのパリティ・エラー、インタラプト・
リクエスト・アクティブ、ビジィ・エラーの各ビット
がクリアされます。読み込まれた値は意味をもちま
せん。

新つくるシリーズ



エレクトロニクスのわかりやすい入門書が欲しいという声をよく耳に
します。しかし、万人に対してわかりやすいというテーマを実現するこ
とは簡単ではありません。

「わかりやすい」ということを実現することはたいへんなのですが、
エレクトロニクスについては「こうやって学べばよいのではないか」と
いう答えがあります。それは、「自分の手で作ってみる」ということで
す。天才は閃きで物事を解明していくことができるかもしれませんが、
凡人にとっては人真似から入るのも合理的です。

ということで用意したのが、この3冊の「新つくるシリーズ」です。
いずれも『トランジスタ技術』誌、および『トラ技ORIGINAL』誌で
掲載され、好評を博した記事のなかから、つくりたくなる記事、つくる
ことを擬似体験できる記事をジャンルごとに再構成しました。真似をし
て体験することが最高の学習になると思いますが、読んでいるだけでも
利用できそうなアイデアをふんだんにカバーしています。

No.1 〈好評発売中〉
つくるツール&測定器

おもな内容●デジタル電圧計
／ファンクション・ジェネレー
タ／カーブ・トレーサ／LCメ
ータ／etc.

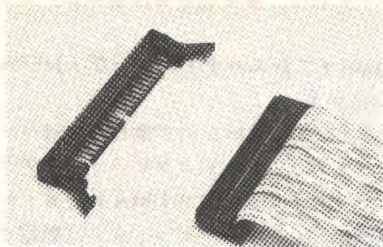
No.2 〈好評発売中〉
つくるオーディオ&ビデオ

おもな内容●オーディオ・アン
プ／サウンド・プロセッサ／ビ
デオ・セクタ／ビデオ・エフ
ェクタ／etc.

No.3 〈好評発売中〉
つくるオリジナル・グッズ

おもな内容●電子ゲーム／キッ
チン・タイマ／電子温度計／電
磁波時計／ニカド電池充電器／
紫外線メータ／etc.

●B5判●160頁●定価1,529円(税込)●



§ 4-3

PC9801用SCSIアダプタの製作

里 和政

PC9801用SCSIホスト・アダプタ・ボードの回路図を図1に示します。パラレルI/O(8255Aなど)ボードを作る程度で製作できます。

ハードウェアの構成

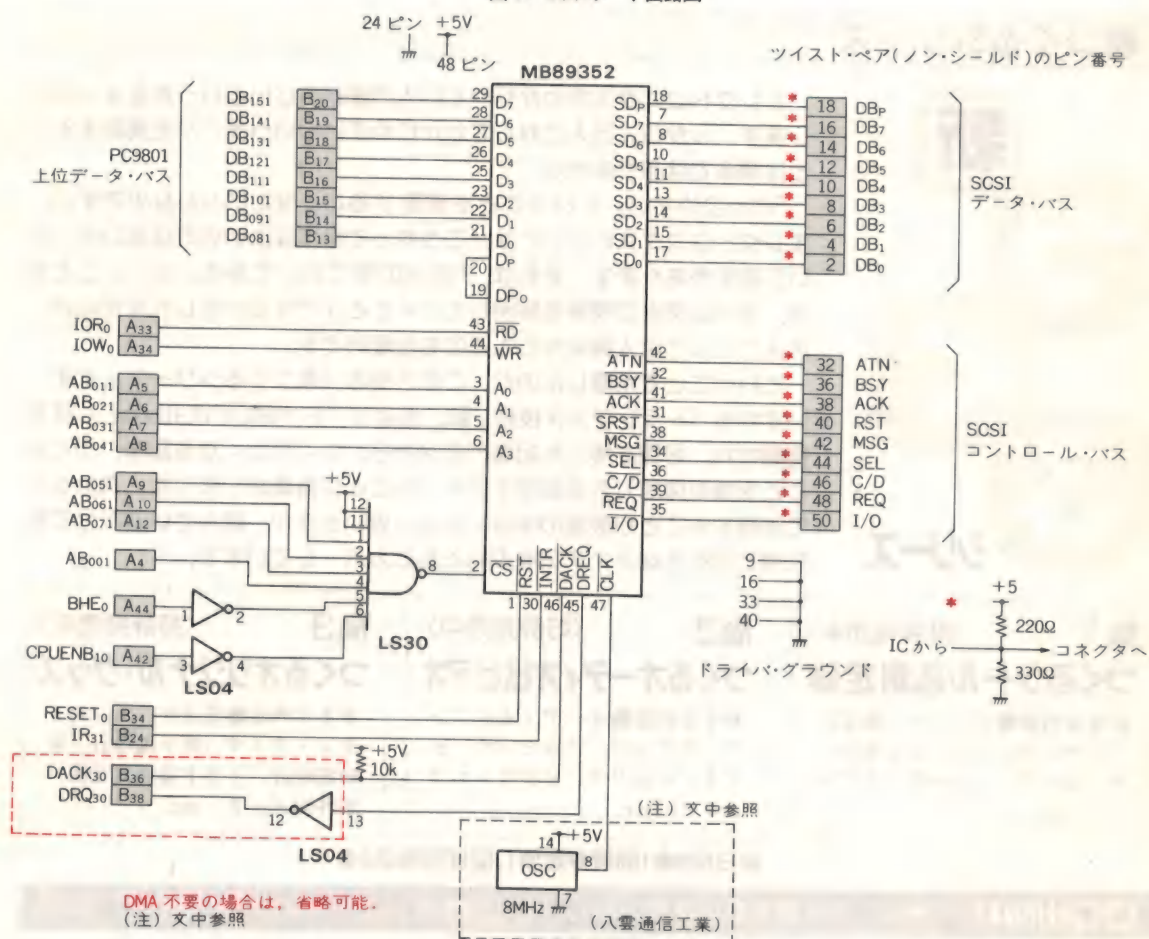
コントローラのI/Oアドレスは、**0E1H~0FDH**までを使用しました。PC9801では、これだけ広いI/O空間は少ないため、アドレスを変更する場合は、

ほかのボードのI/Oと重ならないようにしてください。

また、DMAを使用する場合、PCの偶数のデータ・バス(DB₀~DB₇)に接続する必要があります。そのため、図1のコントローラのデータ・バスをPC9801の偶数のデータ・バスに接続します。このとき、I/Oアドレスは、**0E0H~0FEH**となります。

コントローラからの割り込みは、**8259A(マスタ)のINT₀(IR₃₁)**信号を使用しました。ほかの割り込み信号

〈図1〉 SCSIボード回路図



でも、とくに問題はないでしょうが、スレーブ側的时候は、制御が複雑になります。

コントローラの入力クロックは、最大 8 MHz ですから、PC9801 から直接クロック入力する場合、CPU クロック (SCLK₁) を 8 MHz にします。

48 ピンのソケットが入手困難な場合、24 ピンのソケットを 2 個並べることによって代用できます。

MB89352 のドライバは、不平衡型 (シングルエンド型) で終端抵抗が必要です。シンク電流は、48 mA (0.5 V DC) です。

SCSI 用のコネクタは、50 ピンのノン・シールド型のツイスト・ペア線を使用しました。

SCSIを使用したハード・ディスクの制御

開発用システムは、MS-DOS を使用し、デバイス・ドライバとして作成しました。MS-DOS のデバイス・ドライバには、標準ドライバと拡張ドライバがあります。

前者は、I O . S Y S ファイルに格納され標準装置 (プリンタ、フロッピなど) がサポートされ、システムのローディングに自動的にロードされます。後者は、新しく追加したデバイスを使用したい場合に、容易にシステムに組み込むことができます (マウス、RAM ディスク、拡張ハード・ディスクなど)。

● ハード・ディスクの仕様

今回使用した SEAGATE 社の 40 M ハード・ディスク ST251N は、SCSI を標準インターフェースとしてもっています。表 1 に ST251N の仕様を示します。

SCSI では、ID によって装置を識別するため基板上に ID 設定用のディップ・スイッチがあり、標準は ID = 0 です。

コマンドは、グループ 0、グループ 1 の CDB (コマ

ンド・ディスクリプタ・ブロック) があります。図 2 にコマンドの基本形式を示します。

コマンドの内容はつぎのとおりです。

- (1) グループ・コード：コマンドの記述形式を示す。ここでは、0 または 1。
- (2) コマンド・コード：5 ビットでコマンドを示す。
- (3) LUN (論理ユニット番号)：ドライブのユニット番号。ここでは 0。
- (4) LBA (論理ブロック・アドレス)：MSB ~ LSB の 21 ビットでディスク上のブロック・アドレスを示す。グループ 1 では、32 ビット。通常ディスクは、シリンダ、ヘッド、トラック、セクタによって区分けされているが、SCSI 上ではすべて論理ブロックによって管理されているため、ブロックは連続している。アドレス 0 は、シリンダ、ヘッド、トラック、セクタのすべてが 0 である。
- (5) 転送レングス：ブロックの転送数を示す。0 の場合 256 ブロックの転送となる。グループ 1 では、最大 65536 ブロックの転送が可能となる。
- (6) コントロール・バイト：リンク・ビットは、コマンドを連続して発行する場合 “1” にする。フラグ・ビットは、リンク・コマンドが終了したときのステータス・メッセージのメッセージを示す。フラグ = 0 のとき O A H (LINKED-COMMAND COMP

〈表 1〉⁽⁷⁾
Seagate 社
ST251N の
仕様

トラック数	3,272
シリンダ数	820
回転速度	3600 ± 0.5%
ヘッド数	4
平均アクセス時間	40ms
ビット密度	14,902 BPI
記録方式	2.7 RLL
インターフェース	SCSI
容量フォーマット時	約 40M バイト
セクタ・サイズ	1024, 512 バイト

〈図 2〉
コマンド形式

バイト \ ビット	7	6	5	4	3	2	1	0
0	0	0	0	コマンド・コード				
1	LUN			LBA (MSB)				
2	LBA							
3	LBA (LSB)							
4	転送レングス							
5	コントロール・バイト							

グループ・コマンド部は、0 バイト目の 7 ~ 5 ビット

(a) 6 バイト・コマンド

バイト \ ビット	7	6	5	4	3	2	1	0
0	定義可			—			フラグ・リンク	

(c) コントロール・バイト

バイト \ ビット	7	6	5	4	3	2	1	0
0	0	0	1	コマンド・コード				
1	LUN			—				REL
2	LBA (MSB)							
3	LBA							
4	LBA							
5	LBA (LSB)							
6	—							
7	転送レングス (MSB)							
8	転送レングス (LSB)							
9	コントロール・バイト							

(b) 10 バイト・コマンド

LETE), フラグ=1 のとき O B H (LINKED-COMMAND COMPLETE With Flag)を送信する。

SASI仕様とはコマンドの異なる点があるため、注意が必要です。とくにPC9801で使用しているハード・ディスクの中には、インターフェース上ではSCSI仕様ですが、コマンド・レベルで異なる場合があります。

■ MS-DOSデバイス・ドライバのプログラム

デバイス・ドライバを作成する場合、いくつかの規則があります。デバイス・ドライバの先頭0番地には、**デバイス・ヘッダ**と呼ばれるヘッダがあり、**ドライバのリンク・ポインタ**、**ストラテジ・エントリ・アドレス**、**割り込みエントリ・アドレス**、および**デバイスの属性**を記述します。

ストラテジ・エントリは、コマンド・ブロックのアドレスを受け取ります。コマンド・ブロックはコマンド・パケットとも呼ばれ、コマンド、パラメータの受け渡しに使用されます。

割り込みエントリは、コマンド・コードにしたがって、リード/ライトなどの処理を行います。

デバイスの属性は、ブロック、キャラクタなどを決定します。ハード・ディスクの場合は、ブロック型、NON FAT IDにします。

● BPB部

BPB(BIOSパラメータ・ブロック)は、ブロック・デバイスにおいてファイル構造を定義します。図3にBPBの構成を示します。

次にそのBPBの説明を示します。

- (1) 1セクタのバイト数：論理1セクタのバイト数を示す。

DOSは、これを基準にしてセクタの管理を行います。

- (2) 1ユニットあたりのセクタ数：DOSは、この値からユニットの大きさを決める。

CP/Mでは、データ・ブロックに相当します。1または複数のセクタから成り、2の累乗です。

大容量のディスクの場合この値が少ないと、ディスク全体をカバーできなくなります。

- (3) 予備セクタ数：IPL, システムなどで使用するセクタ数を示す。不要な場合は0にします。

- (4) FATの数：FAT(ファイル・アロケーション・テーブル)は、ユニットの使用状況を示すためのテーブルです。FATが破壊されるとファイルのアクセスが不可能となるため、信頼性を上げるために複数のFATを作成しておきます。通常は2個です。

- (5) ルート・ディレクトリのエントリ数：この数は、ユニット・バイトを32(ディレクトリのバイト数)で割

〈図3〉 BIOSパラメータ・ブロックBPBの構成

フィールド	バイト数
1セクタのバイト数	2バイト
1ユニットあたりのセクタ数	1バイト
予備セクタ数	2バイト
FATの数	1バイト
ルート・ディレクトリ・エントリの数	2バイト
セクタの総数	2バイト
メディア・ディスクリプタ	1バイト
FATのセクタ数	2バイト

ここでいうセクタとは論理セクタを示す

った値の倍数で決めます。

- (6) セクタの総数：全ディスクの論理セクタ数を示す。

- (7) メディア・ディスクリプタ：メディアの交換を行えるデバイスの場合、交換したメディアのタイプを区別するためのもの。

ハード・ディスクの場合、メディアを変更することができないため、とくに指定する必要はありません。

- (8) FATのセクタ数：FATのバイト数は、総セクタ数からユニットの総数を計算します。そのユニット数を1.5(ただしユニット数が4085以上であれば2とする)で割った値の2の累乗です。

■ プログラミング

実際のプログラムにしたがって、コマンドの処理手順の説明をします。

- **初期化**：デバイス・ハンドラをシステムに組み込んだのち、このエントリを実行します。SCSIコントローラMB89352の初期化を行います。

デバイスIDを“7”にし、アービトレーション・フェーズの許可をします。

次に、ディスクに対してテスト・ユニット・レディを出して、ディスクのレディ確認をおこないます。OKならユニット・カウントに1をセットします。

- **データ・リード**：指定されたセクタから指定バイト数のディスク・リードを行います。ディスク上の1セクタは、256バイトでフォーマットしているため、4セクタで論理1セクタとなります。リード中にディスク・エラーが発生したときは、リクエスト・ステータス・センス命令でエラー状態をリセットします。

- **データ・ライト**：指定されたセクタから指定バイト数のディスク・ライトを行います。その他は、リードと同様です。簡略化のためライト・ベリファイも同じにしています。

図4にディスク・コマンドを示します。

図4^(a) ディスク・コマンド

ビット バイト	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	フラグ	リンク

(1) テスト・ユニット・レディ命令

ビット バイト	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	ステータス長							
5	0	0	0	0	0	0	フラグ	リンク

(3) リクエスト・センス命令

バイト \ ビット	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	0
1	0	0	0	LBA (MSB)				
2	LBA							
3	LBA (LSB)							
4	転送長							
5	0	0	0	0	0	0	フラグ	リンク

(5) リード命令

ビット バイト	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	フラグ	リンク

(2) トラック0命令

ビット バイト	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	0
1	0	0	0	FMT DATA	CMP LST	リスト・フォーマット		
2	0	0	0	0	0	0	0	0
3	インターリーブ (MSB)							
4	インターリーブ (LSB)							
5	0	0	0	0	0	0	フラグ	リンク

(4) フォーマット命令

ビット バイト	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	1	0
1	0	0	0	LBA (MSB)				
2	LBA							
3	LBA (LSB)							
4	転送長							
5	0	0	0	0	0	0	フラグ	リンク

(6) ライト命令

● リクエスト・センス命令

ステータス長：センスするステータスの長さを指定する

0：4バイト・ステータス型式

22：22バイト・ステータス型式

27：27バイト・ステータス型式 } 使用しない

ビット バイト	7	6	5	4	3	2	1	0
0	VAL	エラー・ クラス			エラー・ コード			
1	LBA (MSB)							
2	LBA							
3	LBA (LSB)							

VAL：LBAが有効か

(a) 4バイト・ステータス型式

● フォーマット命令

FMT DATA

CMP LST

リスト・フォーマット

特定なトラックのフォーマットを行うときに使用する。

全トラック・フォーマットする場合は、すべて“0”。

インターリーブ：スキューと同様で、フォーマットするセクタ間隔を指定する。0でスキュー“1”。

各フェーズのコントロール手順

セレクト・フェーズは、ディスクIDをセットしてセレクト・コマンドを発行します。このとき自動的にアービトレーションが実行されます。

イン・アウト・フェーズは、転送フェーズにしたがってフェーズ・コマンドをコントローラにセットします。転送の方向は、フェーズによって実行します。図5のフローチャートを参照してください。

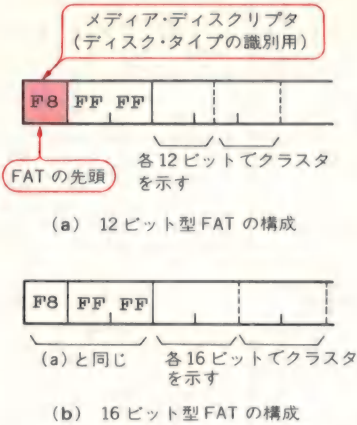
各コマンド実行後エラーが発生したときには、リクエスト・センス命令によってエラー状態を解除します(リスト1)。

図 ディスクのフォーマット

デバイス・ドライバがあっても、MS-DOS用にディスクがフォーマットされていないと実際に使用することはできません。物理的なフォーマットは、ディスクのフォーマット命令でおこないます。MS-DOSではFATとディレクトリ部のフォーマットだけで使用することができます。

図6にFATのフォーマットを示します。FATの先頭は、メディア・ディスクリプタがあり、これでディスクのタイプを識別します。ハード・ディスクでは、0F8Hとします。これが不正なIDであれば、

〈図6〉 FATの構成



メディア・ディスクリプタの種類

0F8H : ハード・ディスク
 0F9H : 640K バイト・ディスク, 1トラック9セクタ
 0FBH : 640K バイト・ディスク, 1トラック8セクタ
 0FCH : 160K バイト・ディスク, 1トラック9セクタ
 0FDH : 320K バイト・ディスク, 1トラック9セクタ
 0FEH : 256K バイト・ディスク, 1トラック26セクタ
 160K バイト・ディスク, 1トラック8セクタ
 1M バイト・ディスク, 1トラック8セクタ
 0FFH : 320K バイト・ディスク, 1トラック8セクタ

CHKDSKコマンドでエラーが発生します。

リスト2にフォーマットのプログラムを示します。
 このプログラムでのフォーマットは、フォーマット・
 コマンド標準の1セクタ256バイトにしています。

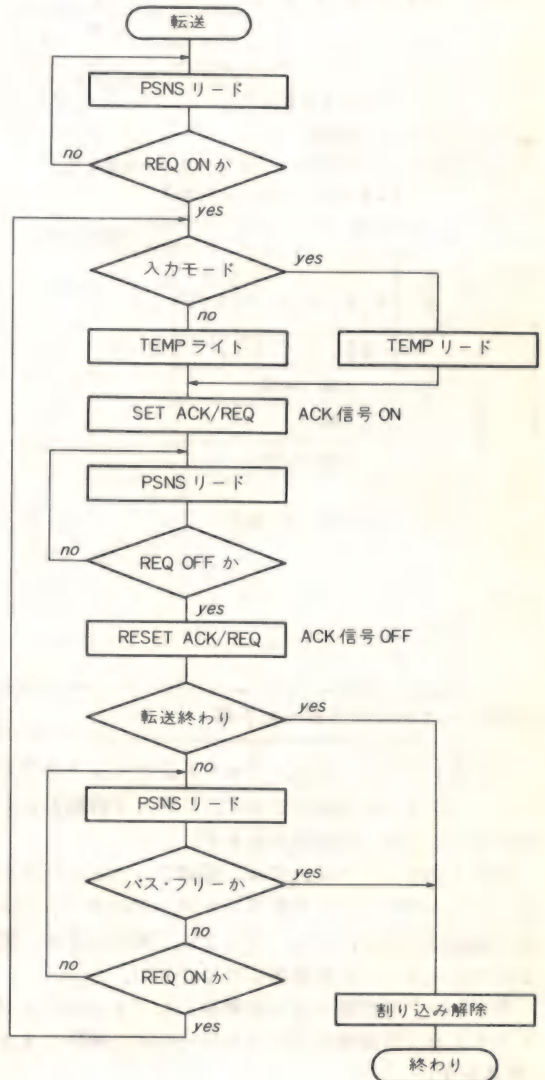
● リセレクトの手順について

SCSIは、前にも述べたように複数のデバイスを接続することができるため、1台のデバイスがバスを占有することは効率の低下のもとになります。そのために、デバイスがバスを使用する場合のみ占有することができます。

この方法は、デバイスによって異なる場合がありますが、SEAGATE社の仕様で説明します。

- (1) セレクト時にATN信号をONにする。
- (2) メッセージ・アウト・フェーズになり、リクエスト・メッセージとして000Hをターゲットに送信して、ディス・コネクトの要求を行う。
- (3) コマンド・フェーズに移りターゲットにコマンドを送信する。
- (4) コマンド受け付け後ターゲットは、ディス・コネクトを行うためメッセージ・イン・フェーズで04H (DISCONNECT)を送る。ただし、ステータス、メッセージ・イン・フェーズは存在しない。その後、バスは解放される。
- (5) ターゲットがバスを使用する場合、ターゲットからアービトレーションを開始する。このときのイニシエータIDは、最初のセレクト時のイニシエータIDを使用する。
- (6) ターゲットは、リセクションを行い、リセクション後メッセージ・フェーズで70Hを送る。
- (7) 以後、通常処理と同じである。

〈図5〉 転送シーケンスのフローチャート



プログラムの使い方

- (1) コントローラを初期化するために、SCSI_INIT をコールする。
- (2) データ・バッファ用のアドレスを、ESにセグメント、SIにオフセットを設定する。
- (3) アクセスするセクタを、BX: CXに設定する。
- (4) リードまたはライトするセクタ数をDLに設定する。
- (5) ディスク・リードは、READを、
ディスク・ライトは、WRITEをコールする
- (6) リード、ライト時にエラーが発生した場合は、AXに0以外がセットされる。

```
*****
: D I S K   リード
*****
read:
    mov     ax,ds
    mov     es,ax
    mov     si,offset disk_buff
    mov     bx,0
    mov     cx,0
    mov     dx,8
    call    disk_read
    ret

*****
: D I S K   ライト
*****
write:
    mov     ax,ds
    mov     es,ax
    mov     si,offset disk_buff
    mov     bx,2
    mov     cx,0
    mov     dx,8
    call    disk_write
    ret

*****
: S C S I   D I S K   インターフェース
*****
: *
: *
: *   ES: SI   =   リード・バッファ・アドレス
: *   BX      =   リード・セクタ (MSB)
: *   CX      =   リード・セクタ (LSB)
: *   DX      =   リード・プロック数
: *
: * disk read
: *
: *
: *   disk_read:
: *       mov     ssci_code,08h   ; read command
: *       and     bl,01fh         ; lun mask
: *
*****
```

```
*****
: *
: * scsi haed disk read/write program
: *
: * ver 1.0
: *
*****
data    segment
: *
: * scsi work area
: *
: * scsi_buff    equ     $
: * scsi_code    db      0
: * scsi_lba0    db      0
: * scsi_lba1    db      0
: * scsi_lba2    db      0
: * scsi_len     db      0
: * scsi_ctb     db      0
: * scsi_stat    db      0
: * scsi_msg     db      0
: * req_stat     db      0
: *              db      4 dup(?)
: *
: * command code
: * block no.
: * block no.
: * block no.
: * block length
: * control byte
: * status area
: * message area
: * i/o mode flag
: *
: *
: * disk_buff    db      256*8 dup(?)
: *
data    ends
CODE    SEGMENT byte
ASSUME  cs:code,ds:data,es:data
```

```

mov     mov     scsi_lba0,bl      ; i/o block no. set
mov     mov     scsi_lba1,ch      ; lba lsb set
mov     mov     scsi_lba2,cl      ; read length (block count)
mov     mov     scsi_len,dl       ; control byte
call    call    scsi_ctb,0        ; select error
jnz     jnz     disk_error        ;
call    call    cmd_send          ; offset set
jnz     jnz     disk_error        ; len * 512
mov     mov     bx,si             ; data in phase
ch,scsi_len
cl,cl
al,1
mov     mov     scsi_io          ; read sector
call    call    disk_error        ;
jnz     jnz     disk_status:

disk_status:
mov     mov     bx,cx
es,bx
mov     mov     bx,offset scsi_stat
mov     mov     cx,1
mov     mov     al,3
call    call    scsi_io          ; status read phase
jnz     jnz     disk_error        ;
mov     mov     bx,offset scsi_msg
mov     mov     cx,1
mov     mov     al,7
call    call    scsi_io          ; message in
jnz     jnz     disk_error        ; error
cmp     cmp     scsi_stat,0      ; normal end ?
jz      jz      disk_end

call    call    disk_req_sen      ; error request sense
jnz     jnz     disk_error        ;

disk_end:
xor     xor     ax,ax
ret

disk_error:
mov     mov     ax,-1
ret

;* disk write
;* ES:SI = ライト・セクタ・アドレス
;* BX = ライト・セクタ (MSB)
;* CX = ライト・セクタ (LSB)
;* DX = ライト・ブロック数

;* disk write
;* scsi_code,0ah ; write command
;* bl,01fh ; lun mask
;* ; i/o block no. set
;* scsi_lba0,bl ; lba lsb set
;* scsi_lba1,ch ; read length (block count)
;* scsi_lba2,cl ; control byte
;* scsi_len,dl ; select error
;* scsi_ctb,0 ; offset set
;* cmd_send ; len * 512
;* disk_error ; data in phase
;* bx,si ; read sector
;* ch,scsi_len ;
;* cl,cl ;
;* al,1 ;
;* mov scsi_io ;
;* call disk_error ;
;* jnz disk_status:

;* disk sense
;* disk_sense:
;* mov scsi_code,00h ; sense command
;* scsi_lba0,0
;* scsi_lba1,0
;* scsi_lba2,0
;* scsi_len,0
;* scsi_ctb,0 ; control byte
;* scsi_select ; select error
;* disk_error ; command execute
;* cmd_send ;
;* disk_error ;
;* disk_status
;* jmp

;* disk request sense
;* disk_req_sen:
;* mov scsi_code,03h ; sense command
;* scsi_lba0,0
;* scsi_lba1,0
;* scsi_lba2,0
;* scsi_len,0
;* scsi_ctb,0 ; control byte
;* scsi_select ;

```



```

mov     [bp],cx
and     al,07h
out     pctl,al
mov     mode,al
mov     cx,0

phase_chk:
in      al,pns
test    al,80h
loopnz phase_chk
jz      scsi_io_err

mode_chk:
test    mode,1
jnz     in_phase
mov     al,es:[bx]
inc     bx
out     temp,al
short set_ack

; input phase
in_phase:
in      al,temp
mov     es:[bx],al
inc     bx

set_ack:
mov     al,0e0h
out     scmd,al
mov     cx,0

out_wait2:
in      al,pns
test    al,80h
loopnz out_wait2
jnz     scsi_io_err
mov     al,0c0h
out     scmd,al
dec     word ptr [bp]
jz      phase_exit
mov     cx,0

out_wait3:
in      al,pns

; save move length
; mask
; transfer phase set
; transfer mode set i/o
; time out counter
; scsi status read
; req ?
; no
; time out
; input phase ?
; yes
; data byte get
; send
; data input
; data set
; set ack/req
; req ?
; time out
; reset ack/req
; move counter down
; end of data

code
ends
end

```

```

or      al,al
jz      phase_exit
test    al,80h
loopnz out_wait3
jz      scsi_io_err
short mode_chk

phase_exit:
in      al,ints
out     intr,al
xor     ax,ax
jmp     short scsi_exit

scsi_io_err:
mov     ax,-1
or      ax,ax

scsi_exit:
pop     cx
pop     bp
ret

;*****
; scsi controller init
;*****
scsi_init:
mov     al,7
out     BDID,al
xor     al,al
out     SCMD,al
out     PCTL,al
out     TCH,al
out     TCM,al
out     TCL,al
out     TEMP,al
mov     al,100000000B
out     SCTL,al
mov     al,00010000B
out     SCCTL,al

; RESET
; ARBITRATION ENABLE

```



```

mov     cx,1
mov     al,3
call    scsi_io      ; status read phase
mov     bx,offset scsi_msg
mov     cx,1
mov     al,7
call    scsi_io      ; message in
pop     bx
pop     es
xor     ax,ax
ret

disk_error:
ret

; * disk write all e5h fill
; * cx : write length (block count)
; * bx : start sector
; *
disk_write:
push    cx
call    scsi_select
pop     cx
jnz     disk_error   ; select error
mov     scsi_code,0ah ; write command
mov     scsi_lba0,0
mov     scsi_lba1,bh
mov     scsi_lba2,bl
mov     scsi_len,cl
mov     scsi_ctb,0
mov     cmd_send
    cl,1
    shl     ch,cl
    xor     cl,cl
mov     bx,offset write_buff
mov     al,0          ; data out phase
call    scsi_io      ; write sector
jmp     short disk_status

; * scsi command send phase
;
cmd_send:
push    es
push    bx
push    cx
mov     bx,cs
mov     es,bx
;
; BUS DEVICE ID
; RESET
;

```

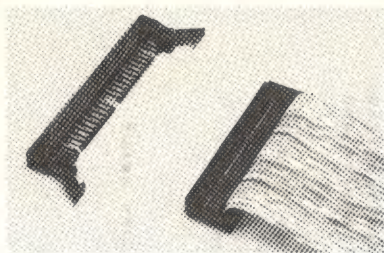
```

mov     bx,offset scsi_buff
mov     cx,6          ; 6 byte send
mov     al,2          ; command phase
call    scsi_io
pop     cx
pop     bx
pop     es
ret

atn_wait:
in      al,psns
and     al,07h
cmp     al,6
jne     atn_wait
mov     al,40h
out     scmd,al
ret

; atn reset
;
; *****
; * scsi control
; *****
;
BDID EQU OEOH
BDID+2 EQU BDID+2
BDID+4 EQU BDID+4
BDID+8 EQU BDID+8
BDID+10 EQU BDID+10
PSNS EQU PSNS
BDID+12 EQU BDID+12
BDID+14 EQU BDID+14
BDID+16 EQU BDID+16
BDID+18 EQU BDID+18
BDID+20 EQU BDID+20
BDID+22 EQU BDID+22
BDID+24 EQU BDID+24
BDID+26 EQU BDID+26
BDID+28 EQU BDID+28
;
; scsi controller init
;
scsi_init:
MOV     AL,7
OUT     BDID,AL
XOR     AL,AL
OUT     SCMD,AL

```

§ 4-4

NCR53C80評価用ボード の製作

清水哲夫

SCSIの動作および実際の転送速度などを調べるために、評価ボードを製作しました。主な仕様は以下の二つです。

- (1) DMA転送可能なワンボード・マイコンとする
- (2) SCSIポートには、市販されているSCSIコントロール用LSIを使用する

● CPUにはBASIC内蔵の8052を用いる

CPUには、インテル社の8052AH-BASICを用いました。このチップは、**ワンチップ・マイコン8052の内蔵ROMに実数型BASICインタープリタをROMに焼き込んだものです。**

今回のように、新しい周辺LSIの評価や、何かちょっとしたことをやりたいときにたいへん便利です。RS-232Cのダム・ターミナル(たれ流し端末)を接続するだけで、立派なBASICマイコンとなりますし、EPROM書き込みのソフトも内蔵しているので、書き込み電圧発生回路を付加すれば、作ったプログラムをその場でROMにできます。

また、パワーONで自動的にそのプログラムを起動させることもできるので、ターミナルを必要としない機器組み込み用のプロセッサとして随分と重宝しています。気になるのは、実行スピードと価格です。スピードに関しては、これはもうBASICということで、あきらめるしかなく、速度を要求する部分は、アセン

ブラのルーチンをBASICからCALLして何とかしのげます。

ただし、このオンチップBASICのもう一つの特徴に、**BASICの機能の大部分(合計62個のサブルーチン)を、ユーザの作成するアセンブラ・ルーチンから、システム・コールという形で利用できるのです。**この機能を利用することで、実数の四則演算はもとより、対数などの初等関数、ターミナルとのI/Oなど、全部をアセンブラで記述するには、かなりの仕事量となるものが、簡単にすませられます。使い方の例を図1に示します。

この機能をうまく使うと、BASICの部分は機械語サブルーチンへのCALL文一つで、残りはすべてそのアセンブラのルーチンですませるということも可能です。

価格は、秋葉原の店頭価格で約5000円と、Z80の約300円に比べれば20倍近い価格です。しかし、これもライブラリ・ソフト込みの値段と考えれば、それなりに納得がいくのではないかと思います。

● SCSIコントローラには53C80を使う

SCSIのコントロール用LSIには、NCR社の53C80を使用しました。これは、SCSI用LSIとしては、おそらく世界で最初に量産されたと思われるNMOSタイプの5380のC-MOS版で、オープン・コレクタのドライバ/レシーバまで含んだ、ワンチップのコントローラです。

〈図1〉
BASIC内のライブラリの使用例

```
① MOV  A, #9AH  ;TOS=R2:R0
   CALL 30H      ;ライブラリ・コール
② MOV  A, #1FH   ;ルート演算 TOS=SQRT(TOS)
   CALL 30H
③ MOV  A, #1H    ;R3:R1=INT(TOS)
   CALL 30H
```

〔プログラムの説明〕

- ① R2:R0のレジスタ・ペアで示された16ビット整数を、実数に変換して、TOS(トップ・オブ・スタック)に入れる
- ② TOSの内容をルート演算してTOSに入れる(実数演算)
- ③ TOSの実数をポップして、16ビット整数に変換し、その値を、R3:R1のレジスタ・ペアに入れる

- (1) パスの形態はシングル・エンド
- (2) イニシエータ、ターゲットの両方をサポート
- (3) データ転送速度は、最大1.5Mバイト/s
- (4) アービトレーションをサポート
- (5) パリティの生成およびチェックをサポート
- (6) 同期転送はサポートしない

- (1) プログラムI/O, DMAの両方共可
- (2) マイコンへの割り込み可

回路の説明

CPUは、最高スピードの12MHzで働かせます。これで80%のインストラクションは1 μ sのサイクル・タイムで動きます。

アドレス・ライン(A₁₅~A₀)と、データ・ライン(D₇

すなわち、CPUはユーザからのDMAリクエスト信号を、割り込み入力INT₀で受け付けると、その割り込み処理ルーチン内で、ポート1の第6ビット(DMAACK)を“L”にします。そして、自分自身は、INT₀信号が解除される(=“H”)まで、ダミーのループをまわります。

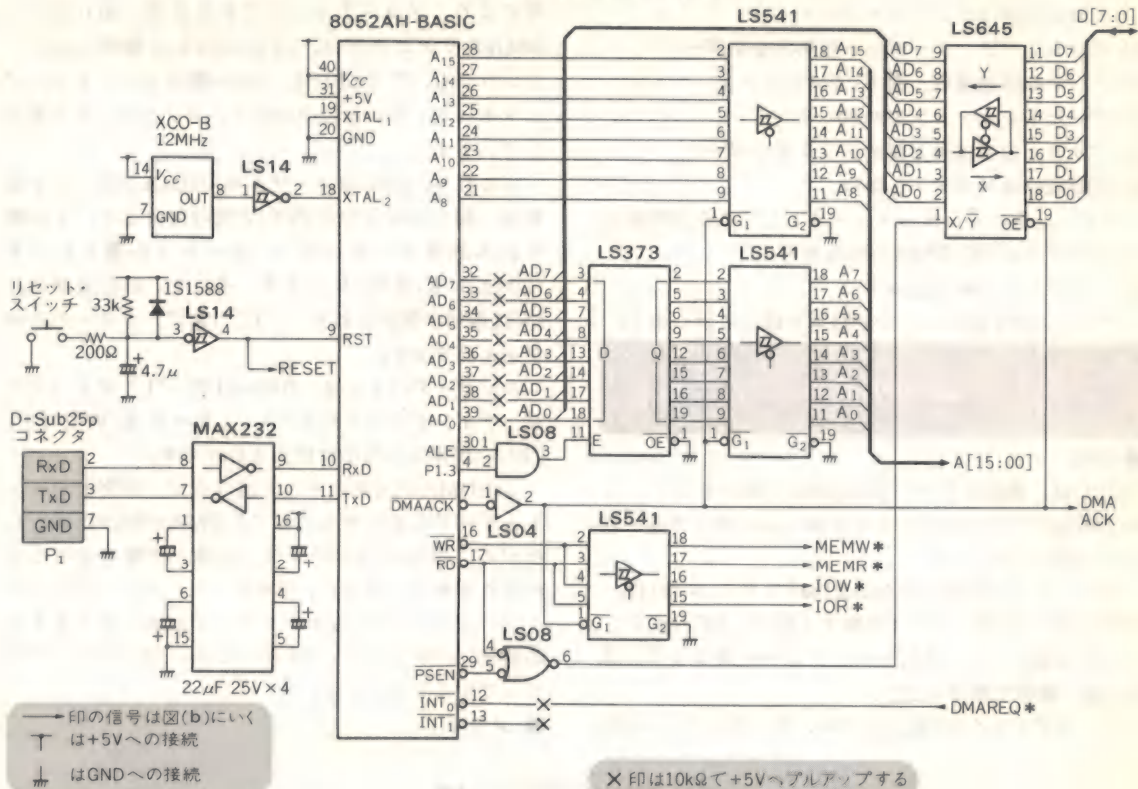
ユーザ・デバイスは、DMAACK="L" によってフローティングになったアドレス、データ、R/W信号を、CPUに関係なく自由に使えるわけです。

このDMAが通常のそれと違う点は、事実上バースト・モードしかできないので、DMAが何か仕事をしながら、同時にCPUのソフトが違う作業をすることができません。さらに、DMAアクノリッジのレイテンシ(リクエストからアクノリッジが返ってくるまでの時間)が割り込み入力を使うため、10 μ sのオーダになってしまうことです。

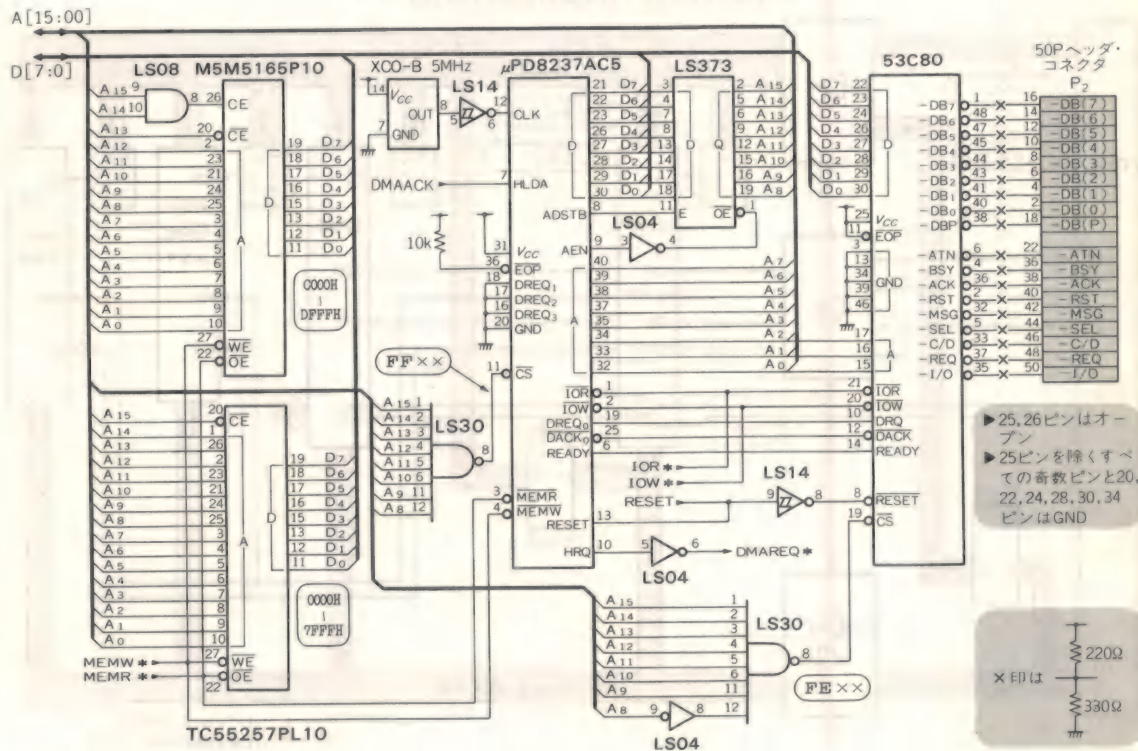
● メモリ

[illegible]

〈図 3 (a)〉 評価用ボードのCPU部の回路



〈図 3 (b)〉 評価用ボードのメモリ, DMA, SCSI部の回路



メモリは、TC55257PL10が、BASICテキスト用のエリア(0000H~7FFFFH)で32Kバイト、M5M5165P10が、DMA転送用のバッファ(0000H~DFFFFH)で、8Kバイト分用意してあります。

● DMAコントローラ

μPD8237AC5は、8085AファミリのDMAC(4チャネル)です。

Z80 DMAなどと違って、メモリ-メモリ間転送以外は、メモリまたはペリフェラル側が、データ・バスを直接駆動するタイプです。DMACは各々のリード/ライト信号(MEMR, MEMW, IOR, IOW)の制御に徹します。電源は+5V 3A程度を用意します。

このワンボード・マイコンは、ターミナル(もちろんTERMモードにしたパソコンでもよい)を接続するだけで、BASICが動作します。ターミナルとの接続は、**RxD, TxD, GNDの3本だけ**のRS-232Cですので、RS-232Cのほかのコントロール・ラインの制御を必要とするターミナルは注意が必要です。

また、XON/XOFFによるハンドシェイクも行えません。データ・ビット長は8、スタート、ストップ・ビットは共に1です。ボーレートは、オート・ボーレートかつプログラマブルですが、ハンドシェイクなしであることを考えると、2400bpsぐらいがちょうどよいかと思います。

動作の確認

製作が終了したら、いよいよ電源ONです。CPUは、RST信号がリリースされたあと、メモリがどれだけ実装されたかをチェックするルーチンを通してから、ターミナルからのスペース・キー待ちとなります。

● ターミナルから動作をチェックする

ここでスペース・キーを1回たたくと、画面に次のようなメッセージが表示されます。

```
*MCS-51(tm)BASIC V1.1*
READY
>
```

最後の“>”記号が、このBASICのプロンプトです。BASICは、最初のスペース・キーからボーレートを逆算して、ターミナルに合わせるわけです。

この後でボーレートを変更する必要があるときは、スペシャル・ファンクション・レジスタであるRCAP2を、次の式に基づいて変更します。

$$RCAP2 = 65536 - \frac{XTAL}{BAUD \times 32}$$

ここで、XTALは使用した水晶振動子の周波数(単位はHz)で、このシステムでは12000000です。BAUDは、変更したいボーレートです。したがって、ボーレートを9600に変えたいときは、

$$>RCAP2 = 65497$$

とキー・インすれば、キャリッジ・リターンをたたいた瞬間に9600ボーに変わり、それからターミナルを9600に変えればよいのです。

● メモリのチェック

本システムでのメモリは、0000H~7FFFFHと、0000H~DFFFFHにマッピングされます。BASIC上のシステム変数MTOPは、0番地から連続して存在するメモリの最高アドレスを示します。

したがって、MTOPをプリントすると、

```
>PRINT MTOP
32767
>
```

となるはずですが、

次に、0000H~DFFFFHのメモリをチェックします。外部データ・メモリ空間にマッピングされたデバイスは、BASICのファンクションXBY(アドレス)で、リード/ライトできます。このデバイスは、CPUのWRとRD信号でストローブされた空間にあります。8052には、外部プログラム・メモリ空間というものもあり、こちらはCPUのPSEN信号によってストローブされるリード・オンの空間のことです。

通常のBASICでは、PEEK, POKEに当たりますが、一つのオペレータで使えたほうが便利です。したがって、

```
>PRINT XBY(0000H)
255
>XBY(0000H)=0
>PRINT XBY(0000H)
0
>
```

のような、ライト/ペリファイ作業を行うことで、メモリのチェックができます。これは何もメモリに限ったことでなく、外部データ・メモリ空間にマッピングされたデバイスは、すべてこの方法でリード/ライトできます。

これがことのほか便利で、筆者などは、とっかえひっかえ、マイコン周辺LSIと呼ばれるデバイスを接続し、この方法でチェックしています。まず、簡単なI/OプログラムをBASICで組んで基本動作を確認し、最後に、その部分を全部アセンブラにして高速化を図って、一件落着といったことをよくやっています。

なお、上の例で注意したいことは、16進数の書き表し方です。最後にHを付けるだけで16進数の意味になります。

(例) 10H=16(10進)

ただし、最初のキャラクタが、A~Fで始まる16進数は、変数と区別するために、その上に0が必要です。

(例) 0F1H=241(10進)

それで、0000Hを表すには、00000Hと入力するわけです。

● 53C80のチェック

イニシエータに設定して、SCSIバス上に $\overline{\text{RST}}$ 信号ができれば、OKということにします。

> XBY(0FE02H)=0

> XBY(0FE01H)=80H

最初の文で53C80をイニシエータに設定し、次の文でSCSIバス上の $\overline{\text{RST}}$ を“L”にします。

ですから、ここで53C80の2番ピンが、“L”レベルになっていればOKです。

> XBY(0FE01H)=0

これで、 $\overline{\text{RST}}$ = “H” にもどします。

ソフトウェア

ソフトウェアをリスト1に示します。プログラムの詳細は図4を参照してください。大部分は、SCSIコントローラである53C80のコントロールで占められています。53C80の詳しい使い方はLSIの説明をみていただくことにし、ここでは、DMAまわりについて説明します。

まずCPUの8052ですが、DMAを行うためには、次の二つのステートメントが必要です。

310 DBY(38)=DBY(38).OR. 02H

320 IE=IE.OR. 81H

文番号310では、8052の内蔵RAMの38番地のビット1をセットします。これによって、INT₀からのインタラプトは、DMAREQの意味で解釈されて、7番

ピンから外部のバッファなどを3ステートにする信号DMAACKが出力されます。

文番号320では、INT₀の割り込みをイネーブルにします。

接続実験

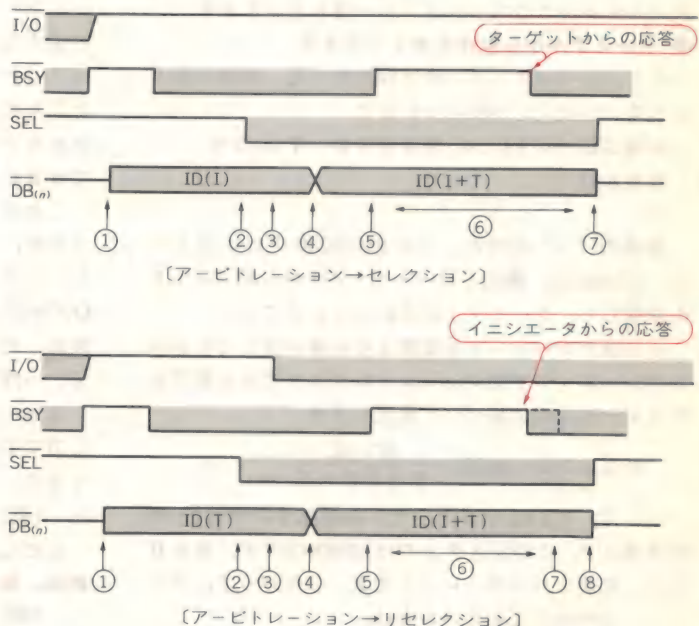
このワンボード・マイコンと、SCSIサポートの3.5インチ20Mバイト・ウインチスタ・ディスク(ドライブはEPSON製、SCSIコントローラ・ボードはDTC製)を接続して、実際にSCSIバスの転送速度や、SCSIの使い勝手などを実験しました(写真1)。

本来ディスクに対してユーザが行おうとする動作は、リードとライトだけです。つまり、何番の論理ユニットの第何セクタから、何ブロック読み/書くという操作です。これ以外のことで必要なことといえば、ディスクのフォーマットができるということぐらいです。これらは、それぞれ、SCSIコマンド・グループ0の

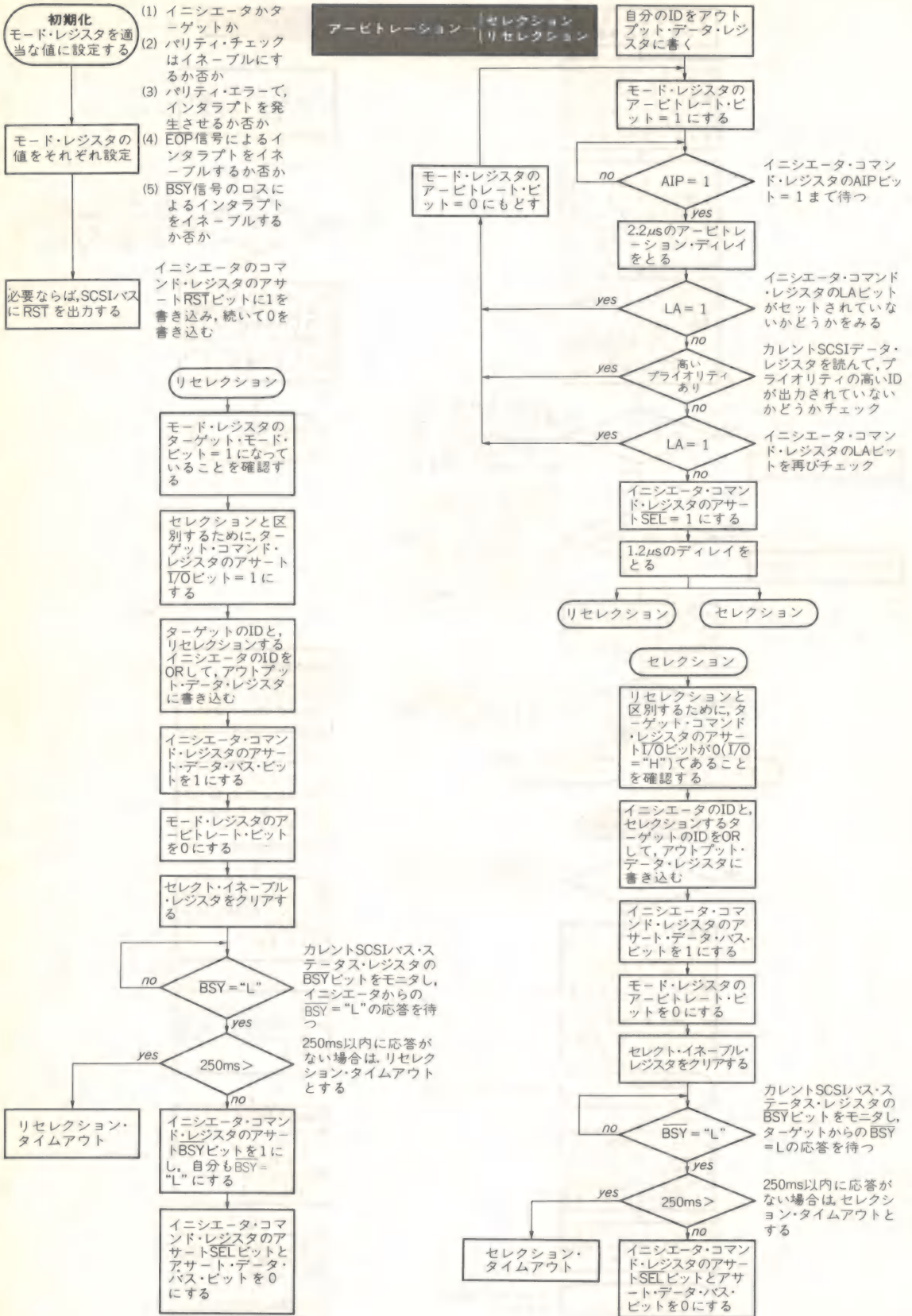


〈写真1〉 接続実験の様子

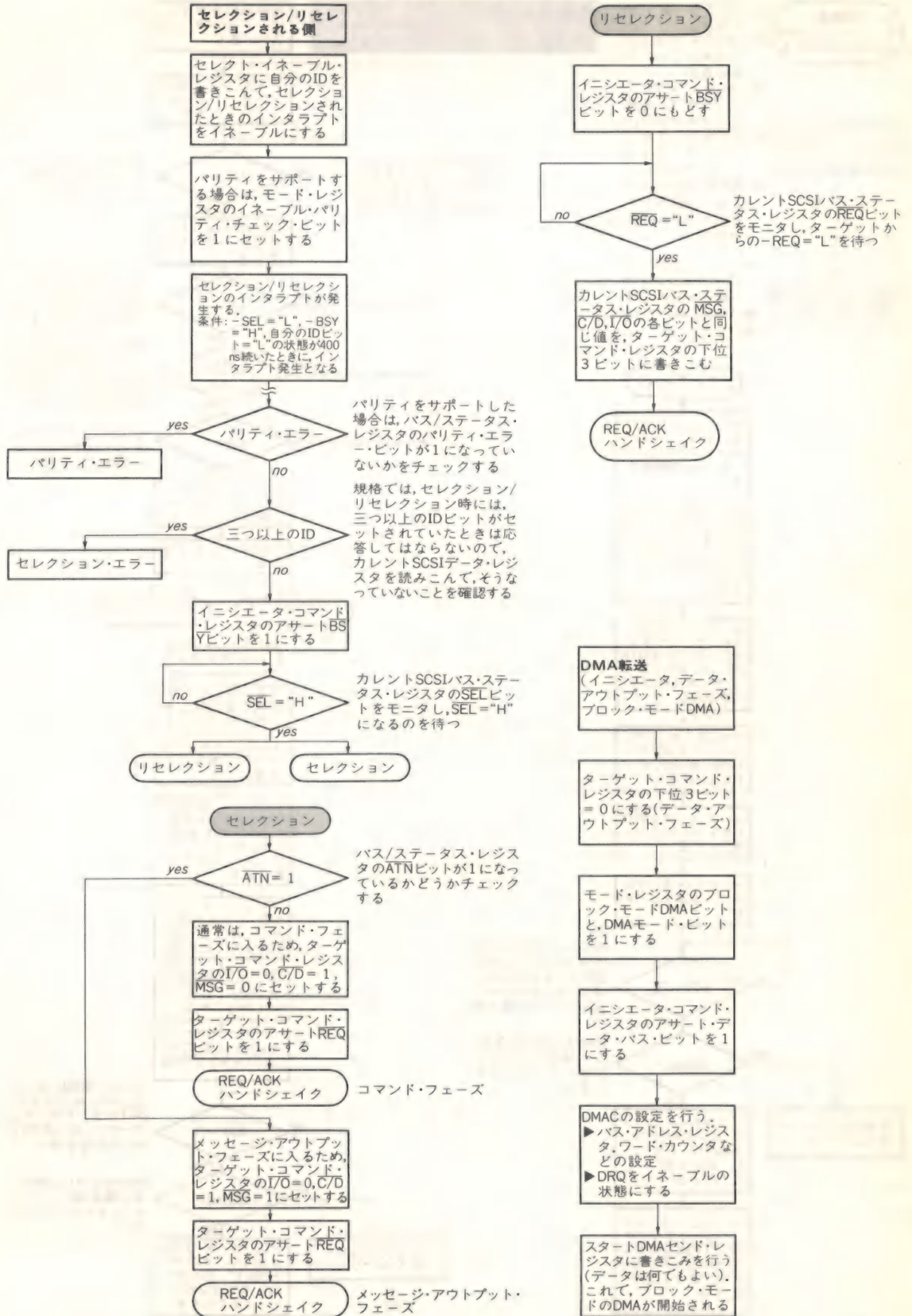
〈図4(a)〉
リスト1の処理の詳細



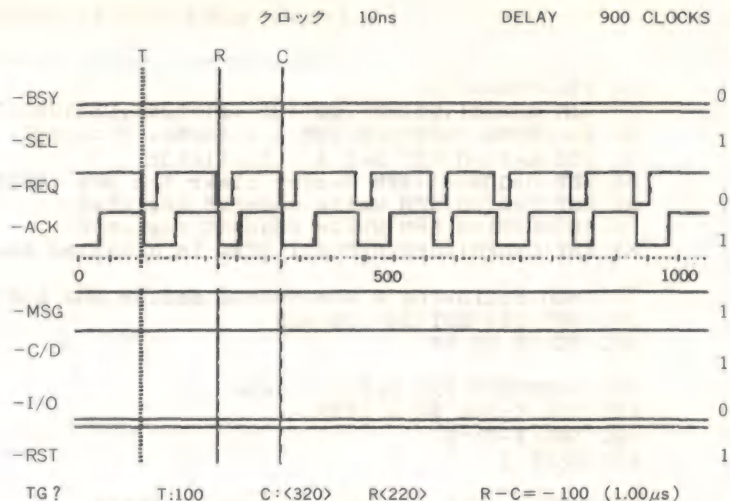
〈図 4(b)〉 リスト 1 の処理の詳細



〈図4(b)〉 リスト1の処理の詳細(つづき)



〈図5〉
実際のタイミングを測定
(データ・インプット・
フェーズ, DMA転送)



READ(コード 08H), WRITE(0AH), FORMAT UNIT(04H)で行うことができます。

実際、エラーが生じない限り、いったんフォーマットしてしまえば、二つのコマンドでディスクとやりとりにできます。今までディスクの管理は、OSまかせにせざるを得なかったユーザも、OSには関係なく自分の管理下に置くことも容易に行えます。

また、転送速度もDMAのおかげで、1Mバイト/sでしており(図5)、BASICで動いていることを忘れさせてくれるほどの快適さです。

参考文献・引用文献

- (1)*MB89352 技術資料 SPC-8704-01P, 富士通。
- (2)*MB89351 SCSIプロトコル・コントローラ仕様書, 富士通。
- (3) PC-9801 ユーザーズマニュアル, 日本電気。
- (4) PC-9801, MS-DOS3.1 ユーザーズマニュアル, 日本電気。
- (5)*PC-9801, MS-DOS3.1 プログラマーズリファレンスマニュアル, 日本電気。
- (6)*Seagate SCSI Interface Manual, November 20, 1986,

36021-001, Revision B.

- (7)*ST251N/ST277N Product Manual, Rev A, Seagate, シーゲートジャパン。
- (8) 清水哲夫: マイコン周辺LSI活用手引き書, 第6回 SCSIの使い方<1> トランジスタ技術, 1987年, 5月号。
- (9)*山本隆夫: SCSIコントローラLSIの動向, データム, 1988年1月号, p.12, CQ出版社。
- (10) ANSI X3.131-1986, SCSI規格書, 日本規格協会。
- (11)*5380-53C80 SCSI Interface Chip Design Manual, NCR (ジャパン マクニクス扱い)。
- (12) John Katausky: MCS BASIC-52 USERS MANUAL, Intel。
- (13) John Katausky: Built-in BASIC interpreter turns controller chip into versatile system core, Electronic Design, December 13, 1984, pp.175~182, Intel。
- (14) 難波秀文: 8052AH-BASICとワンボードマイコンの製作, プロセッサ, Nov 1985, pp.11~25, 技術評論社。
- (15) SCSIバスモニタ OZ101A 技術説明書, 東陽テクニカ。
- (16) DTC-310DB, OEMマニュアル, トーメンエレクトロニクス。
- (17) マルチ・チップ, 1986, pp.170~197, 日本電気。

```

10 XTAL=12000000
20 DIM C0(20),C1(20),D0(128),D1(128),D8(128),D9(128)
30 A=0FE00H:C=0FF00H:REM A = 53C80, C = 8237
40 XBY(A+2)=0:REM Set A = Initiator
50 XBY(C+0DH)=1:REM Master clear the DMA (8237)
60 XBY(C+8)=0:REM Write command register
70 XBY(C+9)=0:REM Write request register
80 XBY(C+0FH)=0FH:REM All DREQ is disabled now

300 REM Following 2 statements define DMA mode
310 DBY(38)=DBY(38).OR.02H
320 IE=IE.OR.81H

400 X=0D0FFH:REM Default data
410 FOR I=255 TO 0 STEP -1
420 XBY(X-I)=I
430 NEXT I

500 PRINT "Hit any key to start program ",
510 X=GET:IF X=0 THEN GOTO 510
520 PRINT CR

600 REM*****
610 REM      Reset
620 REM*****
620 PRINT "Reset!"
630 XBY(A+1)=80H:XBY(A+1)=0:REM Assert RST*

1000 REM*****
1010 REM      Arbitration phase
1020 REM*****
1100 PRINT "Arbitration"
1110 INPUT "Enter initiator ID: ",X0:REM X0=2H
1120 XBY(A)=X0:REM Set ID bits in data out register
1130 XBY(A+2)=1:REM Start arbitration (BSY*=L, IDB(I))
1140 Z=XBY(A+1).AND.40H:IF Z=0 THEN GOTO 1140:REM Wait for AIP
1150 REM Here wait for 2.2 us and check if higher priority exists
1160 Z=XBY(A):IF Z<=X0 THEN 1170:REM Assume no higher priority exists
1170 XBY(A+1)=4:REM Now I'm winning. assert SEL*
1180 REM Here wait for 1.2 us

1400 REM*****
1410 REM      Selection phase
1420 REM*****
1500 PRINT "Selection"
1510 INPUT "Enter target ID: ",X1:REM X1=40H
1520 XBY(A+3)=0:REM I/O* = H for selection
1530 Y=X0.OR.X1:XBY(A)=Y:REM Prepare data bus --> IDB(I+T)
1540 XBY(A+1)=5:REM In order to keep SEL*=L and data bus
1550 XBY(A+2)=0:REM Now reset arbitration --> BSY*=H
1560 XBY(A+4)=0:REM Clear select enable register
1570 REM Now, wait for the X1 target assertion of BSY*
1580 REM And if no assertion within 250 ms. selection fails
1590 Z=XBY(A+4).AND.40H:IF Z=0 THEN GOTO 1590:REM Wait for BSY*
1600 XBY(A+1)=0:REM Deassert SEL* and DATA

2000 REM*****
2010 REM      Now, check what phase the target requests
2020 REM*****
2030 REM 1st, check for BF
2040 X=XBY(A+4).AND.42H:IF X=0 THEN GOTO 2100:REM BF detected
2050 REM Next, wait for REQ* comes
2060 X=XBY(A+4).AND.20H:IF X=0 THEN GOTO 2060
2070 GOTO 2200:REM Now, REQ* comes

2100 REM Here, detected BF

```



```

2110 PRINT "Bus Free":GOTO 500:REM Back to main portion

2200 X=XBY(A+4).AND.1CH:X=X/4
2210 ON X GOSUB 3000,4000,5000,6000,9000,9000,7000,8000
2220 GOTO 2000

3000 REM*****
3010 REM      Data Output phase
3020 REM*****
3100 PRINT "Data Output"
3110 INPUT "Pure data (1) or else (2): ",X
3120 IF X=1 GOTO 3300:REM Pure data
3130 INPUT "Enter # of bytes: ",Y
3140 FOR I=1 TO Y
3150 PRINT "Enter data byte: (",I,"):"
3160 INPUT " ",DO(I):NEXT I

3200 XBY(A+3)=0
3210 FOR I=1 TO Y
3220 Z=XBY(A+4).AND.20H:IF Z=0 THEN GOTO 3220:REM Wait for REQ*
3230 XBY(A)=DO(I):XBY(A+1)=1:REM Set data byte
3240 XBY(A+1)=11H:REM Assert ACK* and data bus
3250 Z=XBY(A+4).AND.20H:IF Z=20H THEN GOTO 3250:REM Wait for NOREQ
3260 XBY(A+1)=0:REM Deassert ACK* and data bus
3270 NEXT I
3280 RETURN

3300 XBY(A+3)=0
3310 XBY(C+0CH)=1:REM Clear 1st/last FF
3320 XBY(C+0AH)=0:REM Enable ch 0 DREQ
3330 XBY(C+0BH)=88H:REM Write mode register
3340 XBY(C)=0:XBY(C)=0D0H:REM Base address = 0D000H
3350 XBY(C+1)=0FFH:XBY(C+1)=0:REM Word count = 255
3360 XBY(A+2)=82H:REM Block mode DMA and DMA mode
3365 XBY(A+1)=XBY(A+1).OR.1
3370 XBY(A+5)=1:REM Now, start DMA Send
3380 REM Here, DMA should have been done
3390 PRINT "DMA finished":XBY(A+2)=0:XBY(A+1)=0:REM stop DMA
3400 RETURN

4000 REM*****
4010 REM      Data Input phase
4020 REM*****
4100 PRINT "Data Input"
4110 XBY(A+3)=1
4120 XBY(C+0CH)=1:REM Clear 1st/last FF
4130 XBY(C+0AH)=0:REM Enable ch 0 DREQ
4140 XBY(C+0BH)=84H:REM Write mode register
4150 XBY(C)=0:XBY(C)=0C0H:REM Base address = 0C000H
4160 XBY(C+1)=0FFH:XBY(C+1)=0:REM Word count = 255
4170 XBY(A+2)=82H:REM Block mode DMA and DMA mode
4180 XBY(A+7)=1:REM Now, start DMA receive
4190 REM Here, DMA should have been done
4200 PRINT "DMA finished":XBY(A+2)=0:REM stop DMA
4210 X=0C000H:FOR I=0 TO 255 STEP 16
4220 FOR J=0 TO 15
4230 PH0.XBY(X+I+J).
4240 NEXT J:PRINT CR
4250 NEXT I
4260 RETURN

5000 REM*****
5010 REM      Command phase
5020 REM*****
5100 PRINT "Command"
5110 INPUT "Enter # of bytes: ",Y
5120 FOR I=1 TO Y

```

```

5130 PRINT "(",I,"):"
5140 INPUT " ",CO(I):NEXT I

5200 REM ----- Y = # of bytes. CO(I) = command data -----
5210 XBY(A+3)=2
5220 FOR I=1 TO Y
5230 Z=XBY(A+4).AND.20H:IF Z=0 THEN GOTO 5230:REM Wait for REQ*
5240 XBY(A)=CO(I):XBY(A+1)=1:REM Set data byte
5250 XBY(A+1)=11H:REM Assert ACK* and data bus
5260 Z=XBY(A+4).AND.20H:IF Z=20H THEN GOTO 5260:REM Wait for NOREQ
5270 XBY(A+1)=0:REM Deassert ACK* and data bus
5280 NEXT I
5290 RETURN

6000 REM*****
6010 REM      Status phase
6020 REM*****
6100 PRINT "Status"
6110 XBY(A+3)=3
6120 Z=XBY(A+4).AND.20H:IF Z=0 THEN GOTO 6120:REM Wait for REQ*
6130 S0=XBY(A):REM Read status byte
6140 XBY(A+1)=10H:REM Assert ACK*
6150 Z=XBY(A+4).AND.20H:IF Z=20H THEN GOTO 6150:REM Wait for NOREQ
6160 XBY(A+1)=0:REM Deassert ACK*
6200 PH0. "Status byte = ",S0:RETURN

7000 REM ----- Message Output -----
7010 REM Message Output is not supported by DTC310DB

8000 REM*****
8010 REM      Message Input phase
8020 REM*****
8100 PRINT "Message Input"
8110 XBY(A+3)=7
8120 I=1:DO
8130 Z=XBY(A+4).AND.3CH:IF Z=3CH THEN GOTO 8300:REM MI and REQ*
8140 IF Z=1CH THEN GOTO 8130:REM Wait for REQ* in MI
8150 REM Here, MI phase is finished
8160 GOSUB 10000
8170 RETURN

8300 D1(I)=XBY(A):I=I+1:REM Read data
8310 XBY(A+1)=10H:REM Assert ACK*
8320 Z=XBY(A+4).AND.20H:IF Z=20H THEN GOTO 8320:REM Wait for NOREQ
8330 XBY(A+1)=0:REM Deassert ACK*
8340 WHILE I<=128
8350 FOR J=1 TO 128 STEP 16
8360 FOR K=1 TO 16
8370 IF J+K-1>128 THEN GOTO 8380 ELSE PH0. D1(J+K-1),
8380 NEXT K:PRINT CR
8390 NEXT J
8400 GOTO 8120

9000 REM ----- IL -----
9010 PRINT "Illegal phase!!!":STOP

10000 REM *** Print out other data bytes ***
10010 REM      Input = I. D1(1 - I)
10020 REM *****
10030 IF I=1 THEN RETURN:REM No other data available
10040 FOR J=1 TO I STEP 16
10050 FOR K=1 TO 16
10060 IF J+K-1>I THEN GOTO 10090 ELSE PH0. D1(J+K-1),
10070 NEXT K:PRINT CR
10080 NEXT J:RETURN
10090 PRINT CR:RETURN

```


●本書掲載記事の利用についてのご注意 — 本書掲載記事には著作権があり、また工業所有権が確立されている場合があります。したがって、個人で利用される場合以外は所有者の承諾が必要です。
また、掲載された回路、技術、プログラムを利用して生じたトラブル等については、小社ならびに著作権者は責任を負いかねますのでご了承ください。

●ご質問はお手紙で — 本書に関する技術的なご質問は、往復はがきか返信用封筒を同封した書簡で出版部あてにお寄せください。著者へ回送し、直接回答していただきます。質問の内容は当該記事を逸脱しない範囲で、できるだけ具体的に明記してください。また、電話やFAXによるご質問にはお応えできませんのであらかじめご了承ください。

トランジスタ技術 SPECIAL

No. 9

©CQ出版(株) 1988

1988年5月1日 初版発行

1997年6月1日 第11版発行

発行人 蒲生良治
編集人

発行所 CQ出版株式会社 ☎170 東京都豊島区巣鴨1-14-2

電話 03-5395-2123(出版部), 03-5395-2141(販売部)

振替 00100-7-10665

(定価は表四に表示してあります)

印刷・製本 三晃印刷株式会社

3端子/チョッパ/フライバック各種レギュレータ ICの使い方

トランジスタ技術編集部 編

電源用 IC活用マニュアル

B5判 160頁

定価1,682円

本書は各種電源用ICのデータの要点と活用方法を網羅しています。

各メーカーのデバイスの中から汎用の電源ICを精選し、タイプ別に電源の設計法や回路例、実測データなどポイントをおさえてわかりやすく解説をしています。

いまや電源システムは、エレクトロニクスの世界では必須であり、その心臓部とも言える役割りをこのICが担っています。技術進歩に伴い、電源ICも高効率で優れたものが開発され、さらに用途別に多種多様の製品が世に出まわっています。

電源システムを構築するための必携マニュアルとして本書をお勧めします。

抵抗、コンデンサ、インダクタ、機構部品の特徴と仕様

薊 利明／竹田俊夫 著

わかる電子部品の基礎と活用法

B5判 184頁

定価1,733円

本書では抵抗、コンデンサ、インダクタ、機構部品の種類とその構造、仕様、特徴をイラストを豊富に使ってわかりやすく解説しています。それに加え、部品の故障率や故障モードなど高信頼設計のための基礎データなどもまとめてみました。ハードウェア・エンジニアには必読の書です。

計測制御の信号処理からセンサ/通信インターフェースまで

トランジスタ技術編集部 編

モジュール化に役立つ実用電子回路集

B5判 160頁

定価1,631円

本書では、あらゆる場面で役立つ、モジュール設計のための回路として、汎用部品でコンパクトに構成した様々な回路を集めました。また設計した回路をより実用的なものにするために、モジュール化設計した回路同士やパソコン、測定器との接続などに役立つ、便利なインターフェース回路も豊富に紹介しています。

DOS/Vマシンのインターフェースを拡張するハードウェア設計

トランジスタ技術編集部 編

IBM PCとISAバスの活用法

B5判 164頁

定価1,835円

本書ではIBM PC/AT互換機の標準入出力インターフェースの仕様をまとめたあと、ISAバスのハードウェアについて詳細に解説しています。さらに、16550Aを使用した拡張シリアル・ポート・アダプタ、高速FIFOを使用したファンクション・ジェネレータ・ボードなど、IBM PC/AT互換機用のISA拡張アダプタ・カードの設計・製作事例を具体的な回路図とサンプル・プログラムを示しながら解説しています。

新つくるシリーズ

●B5判●160頁●各定価1,529円●

No.1 〈好評発売中〉

つくるツール&測定器

おもな内容●デジタル電圧計／ファンクション・ジェネレータ／カーブ・トレーサ／LCメータ／etc.

No.2 〈好評発売中〉

つくるオーディオ&ビデオ

おもな内容●オーディオ・アンプ／サウンド・プロセッサ／ビデオ・セレクト／ビデオ・エフェクタ／etc.

No.3 〈好評発売中〉

つくるオリジナル・グッズ

おもな内容●電子ゲーム／キッチン・タイマ／電子温度計／電磁波時計／ニカド電池充電器／紫外線メータ／etc.

ISBN4-7898-3181-7

C3055 ¥1495E

CQ出版社

定価：本体1,495円（税別）



9784789831819



1923055014954

